Acting to Gain Information

Final Technical Report
Covering Period January 1991 to July 1993

Stanley J. Rosenschein
J. Brian Burns, David Chapman, Leslie P. Kaelbling
Philip Kahn, H. Keith Nishihara, Matthew Turk

Technical Report No. TR-93-03

July 30, 1993

Prepared by:

Teleos Research
576 Middlefield Road
Palo Alto, CA 94301

Prepared for:

NASA Ames Research Center
Moffett Field
CA 94035-1000

Contract Number NAS2-13326

1 Introduction

1.1 Goals of the Project

This project was concerned with agents that act to gain information. By "agent" we mean any computational system that picks up signals from its environment, processes those signals, and generates outputs to control the environment in goal-directed ways. For simple, highly structured environments and tasks, the problem of designing intelligent agents is relatively straightforward and can be handled by existing techniques in control theory and computer science. In cases where the environment or task is complex, new approaches are needed.

Artificial Intelligence (AI) research over the years has developed techniques for modeling complex task domains for which only fragmentary, qualitative domain models are available. Unfortunately, the direct application of those techniques to the agent-design problem is not always possible. The reason for this derives from the computational complexity of inference. In order to exploit qualitative knowledge of the environment, AI has generally adopted the strategy of encoding domain knowledge as symbolic expressions and manipulating these expressions explicitly. While this technique affords great flexibility of expression and narrows the gap between the designer's conception of the domain and the structure of the artifact he is designing, it typically has the unfortunate side effect of requiring slow inferential processing to unwind the encoded knowledge.

In previous work, we have developed models of information and control that are compatible with qualitative modeling while retaining the computational characteristics required for real-time operation [1]. This work has resulted in mathematical models, programming languages, and experimental systems that represent initial steps toward a theoretically grounded design theory for intelligent agents.

To date, however, our main focus has been on modeling how actions affect physical states of the environment; relationships between an agent's actions and information later available to the agent were not explicitly considered. This allowed simplification of the problem, but left open many interesting issues. Treatment of these issues is especially critical in applications where sensory resources are limited and must be allocated carefully while still producing sufficient information to support task behaviors.

The main goal of this project has been to extend previous work by considering problems of active information-gathering explicitly, by developing new techniques for analysis and synthesis, and by exploring aspects of information-gathering in computational perception, learning, and language.

1.2 Summary of Approach and Results

Two concepts are central to the design of intelligent agents: "information" and "action". An agent's intelligence resides precisely in its being able to discriminate situations where differential action is called for and in being able to map those discriminated states coherently over time to overt actions that achieve its goals. In short, an intelligent agent must know what's happening and do the right thing.

Our approach attempts to relate two points of view on how to talk about information and action. The first, inspired by control theory, emphasizes causal feedback paths and the resultant synchronization between agent states and world states. This approach views information as

correlation, and action as a triggered response. The second point of view, inspired by cognitive science and formal models of reasoning, emphasizes symbolic data structures, their interpretation by designers, and their role in means-ends reasoning resulting in rational action. The approach we take in relating these views is to attempt to align them. In other words, we design systems where objective correlation and stipulated denotation line up.

This is especially challenging where the information-update machinery and the information-exploitation machinery are both under the control of the designer. Objective characterizations of the results of action, to the extent that they affect the perception, can only truly be given once the perceptual machinery is fixed. But fixing the perception component prematurely, before an analysis has been done of the space attainable by potential action strategies, may lock out necessary parts of the information space. This circularity seems to argue for iterative refinement as a development strategy

Our theoretical investigations proceeded as follows. First we analyzed agents into their perceptual and action components and identified these with aspects of a state-machine model of a control system. The mathematical properties of each were characterized in isolation and interactions and synthesis techniques, including combinatorial constructions and declarative methods, were investigated. Finally, the interactions that arise between perception and action selection were studied, along with issues of probabilistic uncertainty. These investigations are summarized in Section 2 of this report.

In parallel to these theoretical studies, we considered the phenomenon of active information gathering in a realistic physical domain. We chose visual processing as a challenging test case and developed new approaches to active vision, as described in Section 3.1 Situating our work within the active vision paradigm, we developed a concept of "minimal meaningful measurements" suitable for demand-driven visual processing. We then developed an architecture for interpretation and recognition of visual information that, in state-machine fashion, maintains persistent, meaningful representations of the world. This architecture allows for incremental updates, the representation of partial information, incremental recognition, and the exploitation of statistical information. Experiments with an initial implementation of this architecture are described.

Two other modes of information gathering were studied. The first was learning, which differs from active perception in that only a weak, "universal" model of the task domain was assumed, rather than the kind of detailed models of the physical world that typically underlie perceptual research. The purpose of our investigation was to uncover techniques for coping with some of the combinatorial complexity associated with current reinforcement learning techniques. These results are presented in Section 3.2. Finally, we explored information gathering through explicit linguistic action by considering the nature of conversational rules, coordination, and situated communication behavior. These investigations are summarized in Section 3.3.

1.3 Project Highlights

A Publications

- Leslie P. Kaelbling, completed a book on learning in embedded systems, based on her thesis work, (now published by MIT Press).
- H. K. Nishihara, "Minimal Meaningful Measurement Tools," Teleos Research Technical Report No. TR91-01, June 1991, revised October 1991.
- David Chapman and Leslie Pack Kaelbling, "Learning from Delayed Reinforcement In a Complex Domain," Teleos Research Technical Report No. TR90-11, December 1990.
- David Chapman, "Computer Rules, Conversational Rules," Teleos Research Technical Note. Also to appear in Computational Linguistics.
- Matthew Turk and Stanley J. Rosenschein, "A Vision Architecture for Perceiving Human Action," working document to appear IJCAI-93 Workshop on Looking at People: Recognition and Interpretation of Human Action, Chambery, France, August, 1993.
- Stanley J. Rosenschein, Extended Abstract, IFIP-92, Madrid, Spain, August 1992.
- Philip Kahn, "Building Blocks for Computer Vision Systems," IEEE Expert, to appear.

B Presentations

- NASA Day: Program Status Review for NASA Ames FIA Branch Chief, COTRs, and technical personnel, 6 December 1991.
- AAAI tour presentations. Laboratory tour for attendees of AAAI-92/IAAI-92, 16 July, 1992.
- David Chapman presented a paper on work carried out jointly with Leslie P. Kaelbling on learning from delayed reinforcement in a complex domain to IJCAI-91, Sydney, Australia.
- Keith Nishihara presented a paper on Minimal Meaningful Measurement Tools at the AAAI-91 symposium.
- Stanley J. Rosenschein presented a respondent's position paper at Stanford University on the subject of the semantics of embedded computation.
- Stanley J. Rosenschein, presentations to Robotics seminar, Stanford University.
- Stanley J. Rosenschein, presentations to SIGLUNCH Group, Stanford University.
- Stanley J. Rosenschein, presentation to NASA SOAR Conference, Houston, Texas, 1992.
- Stanley J. Rosenschein delivered a presentation on formalisms for reactive multi-agent systems at a symposium organized by the Electro-Technical Laboratories, Japan, 1991.
- Keith Nishihara and David Chapman made presentations on active vision to the AAAI Fall Symposium at Asilomar, 1991.
- David Chapman, delivered a presentation at the Las Cruces, NM Workshop on the

interaction of perception and language.

- Stanley J. Rosenschein, presentation to IFIP, Madrid, Spain, August, 1992.

C Other

- Several project team members participated in the Active Vision Workshop, Chicago, 1991.
- Keith Nishihara was chosen to head a committee on Hardware (electronic): Computer Architectures, Real-time Image Processing, Foveal Sensor Arrays.
- Stanley J. Rosenschein participated in the Inter-Center Working Group meeting held at NASA Ames, 1993.
- Stanley J. Rosenschein participated in an Air Force Joint Development Laboratory meeting on the subject of real-time planning and control, Utica, NY.

2 Theoretical Considerations

2.1 Motivation and Overview

We are surrounded in nature by organisms that are aware of their environments and that act in ways we perceive as purposeful. These biological systems interact continuously with their physical surroundings, monitoring changes and generating actions through thousands of input and output channels. Even with relatively slow neuronal computing elements, natural organisms are able to discriminate among a staggering number and variety of situations and produce intricate behavioral responses in milliseconds. This capacity for real-time, goal-directed activity has inspired researchers to inquire into the fundamental nature of these phenomena and to attempt to duplicate them in artificial systems. In addition to the obvious intellectual interest of such an endeavor, there are huge practical benefits to be derived: Intelligent monitoring and control "agents" hold the promise of revolutionizing almost every area of human activity, from medical care to factory automation, transportation, and management information systems.

In many ways, conditions are ripe for developing intelligent control technology. At the conceptual level, much is known about the nature of goal-directed activity. The fundamental principles of feedback control systems have been understood since the 1940s, and an impressive body of theory has been developed over the years. Cognitive scientists have studied symbolic information processing in goal-directed agents and have produced semantically rich models of representation structures and means-ends reasoning. On the technological side, the state of the art in hardware components, including sensors, actuators, and computers, has advanced rapidly in recent years, and software systems, including real-time systems and symbolic processing systems, have reached a relatively high level of sophistication.

Despite these advances, a systematic design theory for complex intelligent agents does not currently exist, in part because of the inherent tension between two competing requirements. On the one hand, the designer of a system must be able to express the content of his domain and task model – what is being assumed about the environment and what is desired of the agent – in a form that is understandable and convenient. On the other hand, as in all engineering disciplines, he must produce realizable designs that will operate within resource limits. No existing paradigm has produced exactly the right combination of models and design techniques to resolve this tension for intelligent, real-time agents. The symbolic-reasoning paradigm, while well suited to modeling complex, unstructured domains, has generally fallen short on real-time performance and the demands of interfacing to physical signals. The control-theory paradigm, on the other hand, has been most successful in highly structured domains, especially quantitatively modeled domains, but has not provided adequate tools for treating less structured problems.

Ultimately, we require a paradigm which combines the best of both approaches. The paradigm would offer modeling structures for expressing qualitative, fragmentary, and heterogeneous information about the task domain, as used in declarative AI knowledge-based systems. These symbolic models might then be converted systematically into real-time controllers, just as plant models expressed in the right equational form can be manipulated by control-system designers today to yield optimal controllers. Naturally, because of the approximate nature of qualitative modeling, we would expect the paradigm to guarantee only "approximations" to optimal agents.

Although we are far from having this design paradigm today, some initial steps have been taken. The key has been to generalize the notions of information, action, and goal from the narrow quantitative setting of classical control theory to a richer, more universal setting. This work has been undertaken across a broad front by discrete-event control theorists, computer scientists, and AI researchers [2,3,4]. Work has also been directed at relating the generalized models to symbolic languages and automated deduction in an attempt to reduce the programming burdens

associated with intelligent-agent design [5]. A major focus of these efforts has been to develop robust techniques for trading off semantic richess and computational complexity in the agent's run-time representations.

An interesting variation on the design problem arises when the goal descriptions themselves refer to information, that is, when the agent is described as pursuing a change not in the state of the environment, but in its own internal state. In some sense, this requires the agent to incorporate a model of how its actions, through a causal chain that passes through the environment and its own sensors, circle back to affect its own future information state. As we shall see, this situation raises subtleties that can be illuminated through an analysis of information, action, and internal information flow.

We begin by considering the fundamental nature of physical systems, the phenomena of information and control, and the use of automata as modeling tools, with special focus on semantic issues of representation in embedded systems. This is followed by a treatment of the two main components of an agent: the action-selection component and the perceptual, or information-update, component. In each case, we first model the pure phenomenon (goal-directed action, perceptual updates) abstractly and consider approaches that might ease the synthesis problem in practical design contexts. We then investigate the interactions between perception and action in systems that act to gain information. Finally, we consider probabilistic models that could be used to extend our analysis of active information gathering to situations of uncertainty.

2.2 Information and Control in Physical Systems

Because intelligent-agent design is inherently concerned with causal influences between an agent and its environment, it is natural to begin our discussion of agents at the level of physical systems. All physical systems involve the interplay of space, time, and state. These elements can be modeled in many ways, depending on the purpose at hand, as can the lawfulness of their interrelationships. Classical physics, for example, models space as Euclidean, time as the real line, and state as the distribution of matter and energy in space-time, subject to deterministic laws, often given in the form of differential equations. At the level of everyday experience, we model the natural world in terms of material objects that exist in various macroscopic states and interact with one another in more or less predictable ways. Relativistic and quantum models make use of yet another set of notions.

Systems distributed in time and space can be partitioned, or decomposed, into subsystems with defined interfaces across which causal influence flows through so-called "inputs" and "outputs". The decomposition of central interest to us here is the partitioning of a physical system into two cross-coupled components, the agent and its environment, as illustrated in Figure 2.1. The state of each component evolves over time as a function of its own immediately prior state and input taken from the output of the other component.

The principles underlying systems of this type were first formulated by Norbert Wiener in the 1940s as the theory of feedback control [6]. The key insight is that signals entering the agent and carrying information about the current state of the environment can be used to generate control signals that cause the environment to behave in desired ways. The mathematical models used to describe feedback control systems have been elaborated through several decades of research in control theory and used as the basis for many practical control applications.

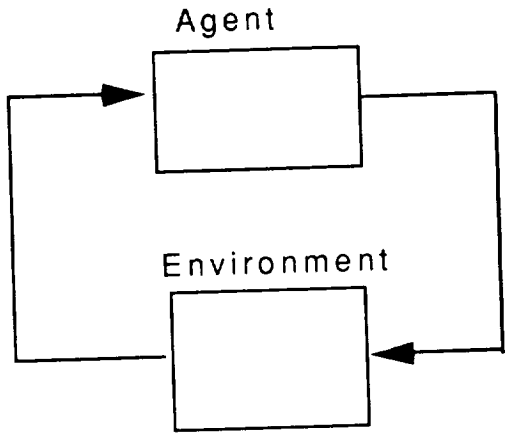

Figure 2.1

The states of an agent and environment in a cross-coupled control system can be modeled continuously or discretely. Continuous models are most useful when the phenomena of interest are very close to physical quantities. When, instead one seeks to capture only coarse or qualitative distinctions among physical events occuring at locations occupied by a system component, it is more natural to employ discrete states, and in such cases, it is also natural to use automata as a mathematical model.

Automata come in numerous varieties, including deterministic, non-deterministic, and probabilistic; they can be finite or infinite and can be decomposed and physically realized in many different ways. Treatment of these issues in their full generality is beyond the scope of this report, but we shall introduce one particular type of model, the non-deterministic automaton. A non-deterministic automaton, A, is defined to be a 6-tuple, $(S,I,O,init,next,out)$, where S, I, and O are the sets of states, inputs, and outputs, respectively. The set *init* , a subset of S, represents possible initial states. The transition relation, *next*, is a subset of $S \times I \times S$, and *out* is the output relation, a subset of $S \times I \times O$. When state s' is a possible successor to state s under input i, we write $next(s,i,s')$. Similarly, when o is a possible output in state s under input i, we write $out(s,i,o)$. In the deterministic special case, the transition and output relations are functions, and *init* , the set of initial states, is a singleton.

The study of control systems whose components are automata, or discrete state-transition systems, has been the domain of discrete-event control theory (for example [4]), and researchers have developed a variety of methods for analysis and synthesis. Most of these techniques employ direct combinatorial methods. That is, they require that the environment automaton be represented as a graph, with all its states and state transitions enumerated explicitly. From this graph, one then computes the desired control automaton. As the number of states of the environment increases, however, the use of direct combinatorial methods becomes problematic and other methods are required.

One category of alternatives involves the use of logical techniques to reason about control behaviors. Logical methods are appealing for modeling ordinary situations because they allow designers to express their knowledge of a domain in convenient linguistic form, a fact at a time. A variety of powerful logical systems have been invented for describing action, change and information. These have been used by philosophers to study commonsense reasoning and by computer scientists to state and prove properties of programs. Typically the syntax of these languages resembles that of first-order logic, but with special operators added to capture the

temporal or other properties of the systems being modeled. By formally describing the environment and task requirements as a set of logical formulas, one can generate provably correct control systems by logical manipulations. Unfortunately, except in very special cases, open-ended reasoning within rich temporal logics is not constrained enough for practical situations, which require a more disciplined, less open-ended approach. Just as direct methods failed on the combinatorics of large state sets of the underlying automata, direct logical methods fail on the combinatorics of inference and search.

In previous work [1], we have attempted to preserve some of the advantages of logical methods while adopting a more structured design methodology. Our approach has been to attach information-theoretic interpretations to system states using automata models, and to do this in a way that allows the objective content associated with representational states in the machine to be specified in a form human designers can understand. States of an automaton can be assigned informational interpretations, expressible in logical notations, by associating those states with environmental conditions with which they are objectively correlated or with which they co-vary. In other words, a machine in state s is viewed as carrying the information that p if the environment always satisfies p when the machine is in state s. When one views the state of the automaton from this informational perspective, it is natural to identify *next* and *out* with their information- and control-theoretic roles, namely keeping information updated and synchronized with the environment, and selecting actions based on current information. These issues are explored more fully in the following sections.

2.3 Pure Action

We begin by considering control in a very simple setting, namely stimulus-response systems that map current inputs to outputs without any dependence on prior inputs. At each instant, the inputs carry information about the immediate state of the environment, but the agent has no internal memory by which to distinguish otherwise similar states through "residues" of past experiences. In the automaton model, the state set of a stimulus-response automaton contains only one state, and inputs are simply passed on to the output relation. This is illustrated schematically in Figure 2.2 (the label "id" designates the identity function on inputs). Although stimulus-response agents are extremely limited, they are complete agents, nonetheless, and constitute a relatively easy-to-analyze leading case.

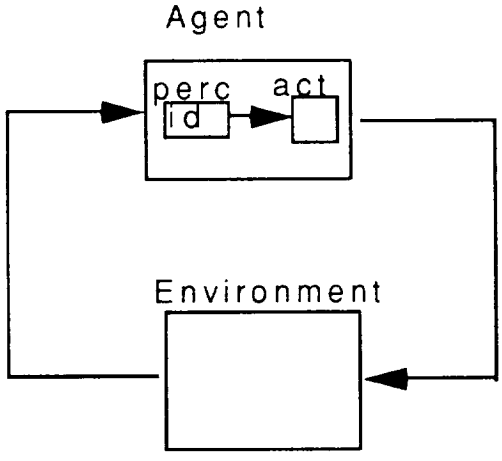

Figure 2.2

By what criteria can a stimulus-response system, or any action-selection system, be judged successful? A natural way to answer this question is to relativize "success" to some stated "goal" specification that is taken to be part of the problem statement. Different families of control problems arise, depending on what is meant by the term "goal."

One dimension of variability in defining goals has to do with fixed vs. dynamic goals. With fixed goals (or design-time goals), action-selection mappings are evaluated relative to the entire trajectory of states they engender. To model an agent as pursuing dynamic goals, on the other hand, assumes some method for defining moment-to-moment variations in what the agent seeks. Even with dynamic goals, the agent is seen as having at least the fixed goal of acting rationally, that is, at each moment selecting actions consistent with its current goals and information. (Interesting subtleties arise in pathological situations where agents might preserve rationality by choosing not to want or not to know.)

Another dimension of variability in goal definition has to do with complexity of the goal and the form in which it is expressed. Goal complexity can vary from that of maintaining simple environmental invariants, to satisfying arbitrary temporal predicates, to optimizing complex numerical evaluation criterion (e.g., maximize throughput while minimizing energy, with complex tradeoffs). Regarding the form of presentation, complexity can vary from a simple enumeration of states, for goals of maintenance, to complex formulas in expressive logical languages, closed under Boolean operations, quantification, and rich temporal operators.

Because in extreme cases, agent synthesis can be arbitrarily intractable, it is not a reasonable objective to try to develop universal solution methods. Rather, it is preferable to develop a methodology for specifying particular action strategies, and to develop an inventory of solved special cases that can expedite agent construction in practical situations. As with goal specification, the specification of action-selection mappings can take many forms, direct and indirect. One family of direct methods includes notations for defining functions of one or more input variables in a suitable language, such as look-up tables (only in very simple cases), functional expressions, circuit descriptions, or data-flow graphs. A related family of methods uses the calculus of relations rather than functional expressions, in some cases with a determinization operator applied as the last step, after the output relation has been composed by applying operations like union, intersection, and restriction to primitive relations. The base-level relations can be represented either enumeratively or in more compact form. In many ways, the symbolic notation used by Gapps [5] provides a declarative overlay for what is at the core a relational calculus, and provides a convenient way of manipulating partially-determined strategies.

Indirect methods define the action-selection mapping by deriving it from some description of the environment and the goal, whether in the form of an explicit combinatorial object like a graph, or in the form of declarative assertions, such as operator descriptions found in classical AI planning systems. To illustrate how a stimulus-response agent can be constructed algorithmically from an explicit description of an environment and goal, we consider the special case of agents that maintain invariants. Although the method illustrated does not scale well with large state sets, it does introduce important concepts and build up intuitions about properties of action strategies.

A stimulus-response agent that maintains invariants can be synthesized as follows:

Let the environment be represented as a non-deterministic automaton *(S,I,O,init,next,out)* which outputs its full state as output. In other words, the output set O is identical to the state set S and the output relation *out* is the identity function on states. Let the goal be represented by G, a subset of S, that the agent is to maintain as an invariant condition.

Working from G, we can compute a kernel set G', with the following properties: (1) G' is a subset of G (i.e., staying within G' guarantees that G is preserved) (2) for every state s in G',

there is an action in I that the agent can take which will keep the environment in G'. G' is calculated by starting with G and computing a sequence $G_0, G_1,...,G_n$ of subsets of G. G_0 is the initial goal set G, and each successive element G_{k+1} in this sequence is obtained by regressing its predecessor G_k back through the next-state relation and intersecting the result with the set G_k. This regression is carried out by finding all predecessor states s such that there exists an action i that has the property that for all states s', if *next(s,i,s')* the s' is in G_k. In other words, we seek states from which we can assure our ability to stay in G_k for at least one more step. The process terminates when a fixed point is reached, i.e., the successive element equals its predecessor. The action strategy is extracted from the fixed point G' by constructing the function that takes each state s in G' to an action i such that all successors of s under i are in G'. If no such action exists, or if G' is disjoint from *init*, the original problem was unsolvable. A variant on this method will work even in cases where *out* is not the identity function.

As mentioned above, this construction does not scale well as the number of environment states increases, and this motivates the use of other representations. Although ordinarily used to handle run-time goals of achievement, the declarative operator descriptions used in AI planning systems, encode the same information as state-transition graphs, and can be used to drive the construction above. Operator descriptions provide a more intuitively interpretable form of expression and can often be manipulated more efficiently because they refer to large subspaces of the state space with terse symbolic labels. Rather than calculating G' through enumeration, operator descriptions allow it to be calculated through symbolic regression. This may be more or less efficient that the alternative, depending on specifics of the problem domain.

The Gapps approach [5] is also based on symbolic representations, but with different semantics. Rather than attempting to describe the objective effects of primitive actions, Gapps provides a language for eliciting from the designer a set of problem reduction statements which express his intuitions about how complete strategies, or policies, can be broken into compositions of simpler sub-strategies, typically specialized to particular types of situation. At the level of primitives the two formalisms converge. As mentioned above, the operational semantics of Gapps is based on a relational calculus.

Up till now we have been assuming that inputs from the environment are sufficiently informative, in that they encode all the world-state information needed to drive action. In cases where less information is available, the inputs to action selection must be derived by accumulating partial information over time, and for this purpose additional machinery is necessary. We refer to this additional machinery as the "perception system" and explore its properties in the next section.

2.4 Pure Perception

As in the case of action selection, it will be useful in approaching perception to begin with a study of the "pure" phenomenon. By pure perception we mean agent-environment systems in which the outputs of the agent have no influence on the environment at all, and the agent is simply a tracking system, or monitor – a passive observer, seeing, but not seen by, the environment. This special type of agent, again, will be of limited practical use but does illustrates the essential features of information extraction. The setup for pure perception is illustrated in Figure 2.3. The lack of influence of the agent on the environment cannot be depicted graphically; it has to do with the details of the *next* relation of the environment automaton and its insensitivity to agent outputs.

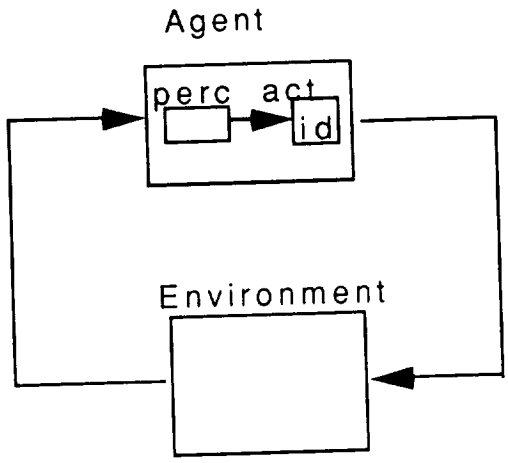

Figure 2.3

The focus in analyzing the perception module is on the kind of correspondence maintained between its states and states of the environment. This correspondence, in fact, is a form of invariant of exactly the type investigated in the previous section, but over the states of the agent-environment pair rather than just the environment. Even when the environment is indifferent to the actions of the agent, it makes sense to ask how the perception component might be designed to maximize the degree of correlation between its states and those of environment, hence maximizing its information.

To see this most clearly, consider an environment, modeled once again as a non-deterministic automaton *(S,I,O,init,next,out)*. What is the maximum amount of information encoded in an instantaneous input? In general, the best we can do is to associate to each input o the set of environment states with which it is compatible (i.e., those s such that *out(s,i,o)* for input o and last agent action i). What is the maximum amount of information about the environment that could be accumulated by the agent automaton over time? Given a rich enough inventory of internal states, a pure perception agent could optimally track the environment by having states isomorphic to the powerset of environment's states, by having its initial state correspond to the environment set *init*, and by having its transition function be that function which maps state S_1 to S_2 precisely when every environment state s' in S_2 is reachable from some environment states in S_1. This might be a cumbersome automaton, indeed, but its tracking behavior would be optimal.

Although as the number of environment states rises, the powerset construction quickly becomes infeasible, it is useful as a thought experiment because much of its value can be preserved through efficient but information-rich approximations. Mathematically, these approximations are homomorphic images of the ideal "powerset" automaton, and thus are consistent with – but not as complete as – that ideal, or optimal, tracker. Nevertheless, these homomorphic images allow useful information to be monitored, while carefully trading off computational space and time, under the designer's control.

One simple approach to constructing homomorphic projections of the powerset automaton is to choose a set of "interesting" or "significant" states in the powerset automaton, and close these under union and intersection. The result is a lattice, which will be a sub-lattice of the powerset Boolean algebra. The construction of the initial state and transition-function of the perception system then proceeds as in the case of the powerset automaton above, but with the "true"

powerset elements approximated by least upper bounds in the sub-lattice. For example, if in the original powerset automaton the transition function maps a state to a successor state that is not an element of the homomorphic-image lattice, the element of that lattice which best approximates the successor state will be returned instead. Thus the lattice transition function "approximates" the optimal transition function and degrades gracefully with the precision of the representation. The lattices themselves would typically be Cartesian products of simpler lattices, with elements that could be represented compactly as parameter vectors.

This technique formed the basis of the Ruler system described in [7]. Ruler takes an approach analogous in many ways to AI planning systems. In Ruler, the environment is described by a set of assertions, including temporal assertions which describe conditions that are either true initially or that will be true in the next state, depending on current conditions. The Ruler compiler synthesizes perceptual machinery (an initial state and next-state function) by chaining together these individual assertions – not with a view toward constructing action sequences, but rather with a view toward computing descriptive parameters in the next state's world model. The use of lattices as the semantic domain of interpretation of the model parameters, along with effectively closing the parameter space under intersection, allows incremental information to be folded in nicely and leads to a compositional methodology for constructing perceptual update mechanisms.

While conceptually adequate for generating provably correct perceptual subsystems, at least for non-probabilistic domain models, Ruler was limited in that it made no special provision for modeling worlds in which objects and their properties and relations were of special importance. This is the case, for example, in visual perception where objects move in and out of view, and a prime form of information to be extracted from the scene concerns the identity of objects and their spatial relations to one another and to the observer. To begin to address domains of this type, we developed an information-update schema we named Percm.

The Percm schema can be thought of as a specialized form of Ruler in which a finite, but shifting, set of objects are being tracked and described. The descriptions are represented as labeled graphs, with node labels representing unary properties of objects, and edge labels representing binary relations between objects. One of the objects is the agent/observer, and the rest of the objects can vary, moving in and out of attentional focus. This scheme bears some relationship to the indexical-functional representations developed by Chapman and Agre for Pengi, but with rigorous correlation-based semantics. The node and edge labels are drawn from a space of data values representing lattice elements, just as in the Ruler case, only now the propositional matrix is fixed (i.e., a fixed conjunction of properties and relations) and the lattice elements are constrained to be of semantic type "property" or "relation", or to be coercible to such values.

The update cycle for this graph is similar to Ruler's, but in the Percm context, fixed background descriptions of the environment are provided not in the form of propositional assertions about world-state transitions, but rather as rules, both temporal and atemporal, for computing object properties and relations. This information is built into a set of operations used to update the relational graph. These operations are:

create	maps an input value to initial object properties and relations inferable from that input;
propagate	strengthens properties and relations among objects x and y by deriving what can be inferred from existing properties and relations between each of x and y and some third object, z (cf. triangulation);

merge	combines descriptions of objects x and y if their properties and relations imply that they are identical;
degrade	maps properties and relations at time t to new values inferable for time $t+1$;
aggregate	creates a new object y whose existence can be inferred from the existence of constituent objects $x_1,...,x_n$ with appropriate properties and relations, and initializes y's description based on descriptions of constituents.

The perceptual system is synthesized by composing and iterating these operations to update the graph of object descriptions, with values again drawn from lattices to obtain gracefully degrading approximations. Because Percm is a finite schema of bounded size, to complete the specification of an instance of the Percm schema, the designer must also define how, in the case of object overflow, objects are to be discarded or withdrawn from active attention.

2.5 Putting It All Together – Acting to Gain Information

The techniques illustrated in the two previous sections can be combined directly to synthesize control systems containing both perception and action components. For instance, using the Gapps approach, one could develop mappings from information states to actions, where the information states are the output of a perceptual subsystem synthesized using the Ruler or Percm methodologies. If there were no interactions among design decisions needed for the two subsystems, the definition of the "information state" of the agent would act as a clean interface, and the combined system would exhibit the intended behavior. In general, however, there are interactions, and in this section we explore the nature of those interactions and how they might be dealt with.

First, there is the issue of how the information-state interface is arrived at in the first place. This question is of concern even in the case of a degenerate perceptual mechanism. One reason has to do with the information-theoretic adequacy of sensor inputs and the computational complexity of mapping sensor readings to actions; the information needed to drive action selection may simply be absent or impossible to extract with available computing resources. Also, certain strategies may require information flows that are impossible to sustain; when choosing action strategies, attention must be given to how actions chosen now maintain the flow of information necessary for distinguishing among future states to be acted on. In AI, this problem often goes under the label of the "knowledge precondition" problem: It is not enough to be in a state when a certain action is appropriate; one must know what that state is and which action is appropriate.

The problem grows more complex when perceptual machinery distills information contained in the sensory input stream, and still more complex when the goal itself pertains to affecting one's own information state. In these cases, the internal structure of the perception module is, from the point of view of the action-selection module, part of some external environment whose dynamic properties are critical to the success or failure of its strategy. Unfortunately, without elaborating the internal structure of the perception module first, statements of fact about this "environment" cannot be made, and hence no valid action strategy can be chosen. In general, action strategies intended to satisfy information goals are only coherently developed in the context of determinate perceptual machinery.

A natural development methodology, then, would be to design the perception module first, choosing conditions to be tracked and defining update circuitry that tracks these conditions in the passive sense introduced in the previous section, but which does not guarantee the input streams that will force it to the right state. After defining this fixed machinery, an action strategy can be

defined, relying on the definition of the perception component as if it were part of the environment. This strategy is designed to cause input streams flowing into the perception component to drive it into the appropriate states and actively makes use of constraints imposed by the previously chosen structure of the perception module. In principle, when perception and action modules are generated from declarative domain descriptions, a single set of facts about the environment should suffice to generate both modules. In other words, Ruler-like state transition rules, combined with operator-description-like action descriptions, contain enough constraints to generate systems that seek information. The Ruler rules generate a perceptual system that maintains, as an invariant, correlations with conditions that the action system needs to test. This approach can involve a search, albeit at design time, for suitable conditions that can be effectively tracked.

In all of these approaches, the output is an automaton with an objective information-theoretic relation to its environments, unlike the usual case in AI when analysis of knowledge pre- and post-conditions has been undertaken in the past using theories that link internal states of agents to their environment only through stipulated semantic-denotation relations attributed by designers somewhat arbitrarily to symbolic data. This distinction is substantial, and it is encouraging that many of the same semantic desiderata that have been pursued in traditional AI planning and representation systems can be achieved in a more mechanistic, and potentially far more efficient, control-theoretic setting.

A final area of complexity, and one which dealt with in some of the empirical research reported in other parts of this report, have to do with uncertainty in natural environments. We have been modeling uncertainty using simple non-determinism. While this allows designers and machines to avoid committing to information they do not possess, they are extremely conservative in that they regard all alternative states that are not ruled out by hard constraints to be of equal importance. In real task domains, however, some of those alternatives are far more likely than others, and this fact is essential to the proper exploitation of the information. A model that is midway between deterministic and non-deterministic models is the probabilistic model in which state transitions, under a given input, are described by probability distributions. A natural mathematical model for such systems is the Markov process, which has been studied extensively by applied mathematicians.

The difficulty in using probabilistic models together with the symbolic techniques described above have to do with the nonmonotonicity of probabilities, which leads to non-compositionality of the design technique. By conditioning on further evidence, the probability of a proposition can, in general, be reduced or increased. This means that a designer cannot, in general, define a module of the perceptual component, prove a strong statement about the semantics of its outputs, and then proceed to use that module together with other modules; conditioning on the joint states of the modules may completely undermine the intended semantics of the first module. Furthermore, the action strategy embodied in the action-selection component is integral to the definition of the probabilistic state-transition matrix of the entire system. Just as before when we could not, in principle, define an action strategy beforeproviding a fixed definition of the perception component, here we cannot define the perception component without constraining action first. The apparent circularity only points to the fundamental need to consider the agent as an integrated whole; the behavior of the entire system – agent plus environment – is determined only when all the boundary conditions have been specified. Interim constraints and incremental refinement may be useful, but must be used cautiously, especially when modeling domains probabilistically.

One simple conceptual extension to the Ruler and Percm approaches allows probabilistic representation to be incorporated smoothly into the overall framework. When designing lattices to represent information states, one can employ the lattice whose elements are mappings from a given propositional lattice into the interval [0,1] under the natural ordering (a mapping is less

than another if it maps elements less than one another to points in [0,1] less than one another). Finite representations of these elements can be approximated by adding confidence values to parametric representations, and with appropriate adjustments to the update algorithms given in the previous section. The simplest such algorithms would maintain compositionality by adding uncertain information conjunctively until a clash is detected and then reset that parameter to the vacuous proposition *true*, with confidence value 1. The justification for the use of an algorithm of this type could depend on global constraints such as ubiquity of evidence for certain propositions or the unlikelihood of monitoring conflicting conditions for a long period without uncovering evidence of the conflict.

In the research reported in this report, we have emphasized empirical approaches to uncertainty, and the use of learning techniques. Results of these investigations are described in the sections that follow.

3 Specialized Modes of Information Acquisition

3.1 Gaining Information Through Visual Perception

A Background: Active Vision

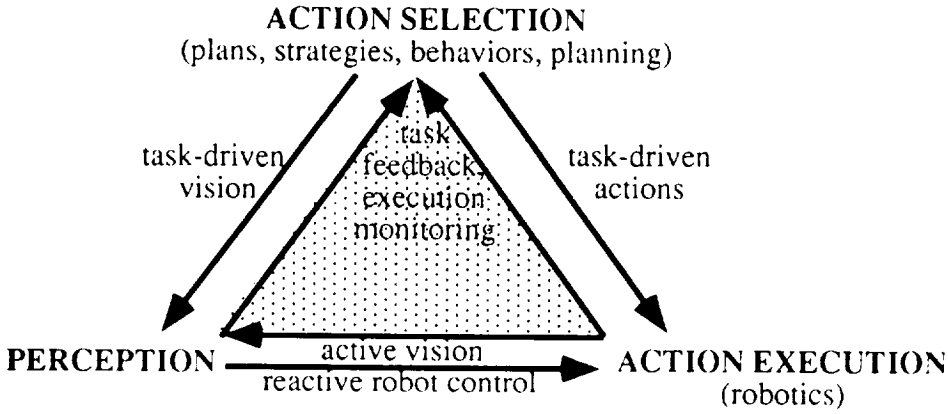

Figure 3.1: Task-Directed Perception and Action Execution

The sophisticated performance observed in biological systems is to a large degree derived from the fluent use of simple and robust measurement capabilities. The task being performed drives how and when sensory actions are to be performed. For example, experienced drivers apply strategies for controlling where and when to monitor the position of the roadway, signs, and other objects; inexperienced drivers are noted for their poor attentional control strategies. Such active vision strategies are a key element of effective sensory robots in complex and uncertain domains.

An active vision system leverages high-speed visual computations, electromechanical control of sensors and mobility devices, and demand-driven control of these devices to achieve a directed goal. A key element of active vision systems is that they can be driven by the overall task and goal achievement systems; for example, a mobile Unmanned Ground Vehicle (UGV) agent can direct visual sensing to support its navigation and task goals.

By its interdisciplinary nature, active vision touches upon a number of fields, such as planning, control, real-time systems and vision. Research specifically targeted for active vision is relatively new. It has included fast recognition algorithms, selective attention mechanisms, movement detection, visual tracking and depth-of-field control [8]. Recently, computer vision conferences (e.g., ECCV, ICCV, CVPR) have included active vision tracks to describe work in this rapidly growing field.

A key strategy used by active vision researchers has been the tight integration of perception, action selection, and action execution. The interaction between perception and action execution has been used to develop real-time active vision systems and reactive robot control. Action selection is the "gamemaster" that can control what activities are pursued and how they are performed; action selection can vary from generating constraints that affect local reactive behaviors (e.g., by modifying desired speed) to more global action sequencing and selection (e.g., task conflict detection and resolution).

Hard perception problems are easier to solve with a good set of primitive measurement capabilities. To be of practical value, a visual measurement capability must provide robust and appropriate measurements in time to be useful at a cost that is not prohibitive. One consequence is an increased focus on demand-driven visual measurements as opposed to the more traditional approach which attempts to carry out a full scene analysis prior to making any use of the information derived. The traditional approach has the disadvantage of being computationally expensive and does not easily support the differing needs of diverse applications.

A demand-driven strategy places a premium on defining a good set of measurements, that is, measurements that are useful, easy to interpret, robust, realizable, and fast. We have defined and implemented an initial set of demand-driven primitives for real-time computation of stereo disparity and visual motion which are noteworthy for their speed, robustness, and wide operating envelopes. Our continuing technology development efforts are aimed at raising performance and lowering the cost of measurement delivery systems.

A.1 Prior Work

Seminal papers in the field of active vision have been written by Bajcsy [9], Burt [10], Krotkov [11], Aloimonos, Weiss, and Bandyopadhyay [12], and Ballard [13]. In 1991, the NSF Active Vision Workshop brought together workers in the field of active vision to assess the state of the field and its future directions and needs [8].

Real-time stereo processing has been the goal of several groups. Matthies at JPL has developed a fast stereo algorithm using sum of squared differences area correlation on a Laplacian Pyramid [14]. His system was implemented using special accelerator boards for doing the convolution with the correlation done on a fast single board computer. That system was able to compute a dense 64 by 60 range map in about 2 seconds and a newer version promises to operate at sub-second rates. It has been actively used for navigation and obstacle avoidance experiments on an autonomous vehicle at JPL. The stereo camera system employed there had a fixed mount on the vehicle looking forward.

The active-vision research at Rochester [15] includes a real-time colored object search and recognition system [16]. At its core, this process matches color histograms to detect the likely presence of a familiar object. The quality of match ranks the set of candidate gazes for further analysis. Additional research at Rochester includes a system for learning selective visual attention mechanisms. Specifically, sequences of fixation points for visual analysis of objects have been learned automatically. Also, methods were developed for fast movement detection, tracking and motion-based figure-ground separation. Coombs demonstrated real-time vergence tracking of people using a fast stereo matching algorithm [17], [18].

Woodfill and Zabih at Stanford have done object tracking using sum of squared differences correlation on raw gray level image data [19]. They were able to do real-time motion tracking of a person walking back and forth using a Heathkit HERO robot for doing pan motions and a Connection Machine for the processing.

Some active vision projects have studied the integration of visual processing and vehicle control. Perceptual servoing, or rapid frame-to-frame tracking of visual targets, has been developed and used by Fennema for UGV navigation [20].

A.2 Visual Attention in Active Vision Systems

Simple perception strategies can be composed to create more sophisticated perception behaviors. The challenge to doing this productively lies in the design of control mechanisms for implementing goal-oriented sequential measurement strategies.

Task behaviors provide isolated robot skills that form the building blocks of competence. In the first systems, robot programs were constructed by fixing the order and function of these building blocks. Robots in manufacturing also typically provide a detailed a priori specification of robot actions. Yet, a given robot task or behavior cannot anticipate all possible occurrences since the ability to model and control the environment is necessarily limited (i.e., open world problems exist [21]). This requires that tasks and the task mix be adapted as new information suggests changing conditions, uncertainty, shifting priorities, and task performance limitations.

The control of sensory robots must consider which tasks should be undertaken in light of competing behavior options, context, and constraints. Humans are quite adept at this. For example, while you are at your desk perhaps doing several tasks at once (e.g., browsing through electronic mail on your terminal while listening to the radio and thinking about what you have to do that day), the phone can ring (noting that somewhere was a process that detected the phone ring and its significance), and some of the current tasks (dealing with email, the radio, day planning) may be abandoned or postponed to service the phone call. In the security and surveillance domain, detection of an unexpected moving object may require halting many current behaviors in order to invoke a behavior(s) that can determine its significance. The effect of contextual information can also be more subtle. For example, the detection of an unexpected moving object may render all noise-producing behaviors less desirable (to achieve greater stealth), and this affects how existing behaviors execute. A key aspect of intelligence is this astute redirection of attention to new information or conditions.

Task-directed control of active vision, robotics, and other embedded real-time intelligent agents operating under uncertainty and functional limitations require:

- real-time behavior execution and control.
- dynamic reconfiguration of current robot activities to conform to changing environment and task requirements.
- information that can be used to prioritize behavioral tasks.
- the application of planners and other mechanisms to focus system attention in light of the current context and the overall mix of current activities.
- flexible and straightforward programming support for modular software development and integration.

B Minimal Meaningful Measurements

B.1 The Intelligent Blind Person

The perceptual information a blind person needs about his environment and the character of aids that prove most useful to him can provide practical guides for research in machine vision. In this section we develop the concept of early vision measurement tools based on insight gained from thinking about the needs of a blind person. There is a close analogy between the demands of building a viable sensing aid for an intelligent blind person and those of building vision front ends for active problem solving machines. We will follow this theme through the definition and development of a demand driven stereo vision system.

In conversations with blind persons, a recurring theme has been the suggestion that to be accepted, a visual aid must be easy to use, useful and cost-effective. Interestingly, there seemed to be little desire for very smart aids. One individual even commented that he was not interested in a machine that could compute the best path down the street for him, he wanted something that

would help him find a lost sock on the bedroom floor and maybe help him match it with other socks in his dresser drawer.

We have intelligent users here and what seems to be called for are aids that such users can operate as tools to accomplish perceptual tasks. Following this line of thought, a desirable perceptual aid ought to recover some basic information and it should have an easy-to-model behavior that is sufficiently rich to allow an expert to use it in creative ways. A blind person's cane is a good example, it has a consistent mechanical behavior and it provides timely information about the presence or absence of physical objects at dynamically selected locations about the operator. The cane "device" has low-bandwidth input and output interfaces to the user – that is, manual pointing control and force, vibration, and sound feedback. This allows it to be managed easily by the blind user while carrying on other parallel activities such as conversation. Furthermore, though simple, the cane has a fairly rich and consistent behavior that fosters the development of expertise in its use. For example, one learns the *feel* of different pavement textures or conditions – slippery or uneven.

Compare the cane with a strawman device – a tv camera driving a 500 by 500 array of vibrating needles attached to a user's back.

While it might sound like a reasonable thing to try, such a device has several shortcomings. First, it is inefficient, providing far more information than any user's back could ever absorb. One estimate is that the human tactile bandwidth supports something closer to what could be presented on an array with just 100 pixels total [22]. This resolution is sufficient to image a single printed character. For all but the simplest imagery, such as reading black and white text, such displays have proven to be useless. The relationship between pixel intensity and physical properties of the environment are too complex and cannot be solved in real-time by even an expert.

We get a quite different result if we strip the strawman device down to a simpler 5 by 20 pixel resolution that better matches the user's tactile input bandwidth and is tailored to display just a single printed character at a time from a hand held scanner. Such a device is much more like the cane and trained users have learned to *read* printed text with it at rates better than 40 words per minute.[1]

The term 'measurement-tool' carries with it the connotation that the device has an easy-to-model behavior that enables its operator to apply it skillfully in specific task domains, much as with the blind person's cane or as an artist comes to know and use a brush. This suggests three guidelines for designing measurement tools:

1. **Simple but meaningful.** The device should make the simplest meaningful measurement possible. A stereo tool, for example, must use a pair of binocular images, but it need not compute a dense range map over an extended field. How much can we simplify the computation without changing the basic character of the measurement?

2. **Easy-to-model.** The device should have a consistent, easy-to-model behavior. If the underlying algorithm has many special case behaviors, it becomes difficult for a user to anticipate that device's behavior in new situations or possibly even in familiar ones.

3. **Informative output.** The device should exhibit a behavioral richness that encourages the learning of strategies for making more specialized measurements with it. For example, simply reporting best estimates of range from a stereo correlation tool would deprive the

1. A device of this sort called the OPTACON is manufactured by Telesensory Systems of Mountain View, California.

user of valuable information about the shape of the correlation peak. In various circumstances, that user might be able to use knowledge of the peak's height, its broadness in vertical disparity, or its bimodality.

B.2 Early Vision Measurement Tools

The vision problem can be usefully divided into broad subclasses of computational modules demarcated by the kind of information at their inputs and outputs [23]. One such class of computations takes information about image intensity at the retina and yields information about visible-surface properties that directly affected those intensities. This class of problems – sometimes referred to as *early vision* – can be further subdivided according to the distinct physical mechanisms by which visible-surface properties can affect intensity images. Some examples are binocular stereo, structure from motion, shape from shading, and material from color [24].

The measurement tool concept can be applied to the study of early vision problems to help us define computational problems that are somewhat different from the problems that are traditionally addressed. Instead of attempting to compute a complete visible-surface description of a scene within a computational module, we will concentrate on single measurements. This distinction can be significant when issues of interaction with higher level knowledge and control are considered.

In the former case, the goal is generally to produce the best visible-surface representation possible from the available input. Unfortunately, it has been very difficult to solve these problems sufficiently well to meet the diverse demands of higher level user processes. It is impossible to anticipate or efficiently support, in a purely bottom up fashion, all the possible uses for the result of an early vision module. Attempts to do so have produced algorithms with complex behaviors which often incorporate many specialized assumptions about the physical scene. What is bad about this is that the application of assumptions is bound up in the early vision modules and out of the control of the user who is often better situated to make decisions about their application. This makes it difficult to bring higher level knowledge to bear on the early analysis.

An alternative proposal is to structure the analysis around making the best possible bottom-up visible-surface measurements at a single location. This makes it is easier to present more precise information about what is known to the user. It can also allow him to better allocate processing resources and interpret basic measurements to accomplish the task at hand with increased efficiently and precision. For example, a range measurement must be made over some finite area, the size of this sensing area can be specified and controlled by the user. In addition, the result can be presented along with a confidence assessment or even as a distribution. This opens up possibilities for straightforward communication between higher and lower level processes. In many instances, intuitive top-down control can be achieved simply through pointing the device and setting the size of its measurement field (zoom).

Measurements such as average range or velocity over areas fit well into this characterization. For example, in stereo matching, a measurement over a small sensing area may fail due to the absence of matchable features. To recover, the user can try switching to a larger measurement window or he can move the smaller window to a slightly different position. In either case, the operator is aware of the changes made and their implications for the measurement. He is in possession of knowledge of the task to be accomplished, he is aware of the measurement difficulty and the character of the degraded information obtained. At the same time this user does not have to know much about the workings of the measurement algorithm itself. The tool can be treated as a black box so long as it has a simple and consistent behavior.

A tool of this sort should also open up opportunities for enhanced resolution through manipulation of the tool. For example, a low spatial resolution range probe can be swept across an extended straight edge while its output is monitored for a change, to get a hyper-resolution reading of its spatial position. It may be that the sophisticated performance observed in biological systems can be explained to a large extent by strategies like this involving the fluent use of relatively simple measurement tools.

B.3 Demand Driven Stereo/Motion

To be of practical value, a visual measurement tool must be capable of providing robust and appropriate measurements in time to be useful, at a cost that is not prohibitive. To accomplish this, we have focused on demand-driven visual measurements. The first class of computations studied extensively in this context have been image matching algorithms applicable to stereo range finding and optical flow field measurement. We have developed a computational theory for measuring stereo and motion disparity that is consistent with the measurement tool objectives and we have had some success at demonstrating the validity of that model for biological systems. We have also developed a practical real-time hardware implementation described later in this section.

a Simple Area-Based Stereo Measurement

Much of the work on stereo vision algorithms in the past several decades has been guided by the remarkably clear percept of *filled-in surfaces* one experiences when looking at random dot stereograms even when the dots are sparse [25]. To make matters more challenging, these same stereograms exhibited crisp depth edges, suggesting that the stereo process was doing its work at very fine spatial resolution. This insight led to a flurry of research activity in the 80s on developing surface interpolation techniques capable of taking sparse stereo measurements and *filling them in* to make a smooth disparity map with sharp discontinuities like those we *see* in the stereograms. But this is not the only way to look at this problem and it is not clear that our perceptual experience with random dot stereograms implies the existence of an explicit filled-in representation in the human system.

We advocate a counter position: No explicit interpolation is required to explain perceptual experiences such as *filled-in* random dot stereograms. Furthermore, stereo perception has surprisingly poor spatial resolution but exceedingly good noise tolerance [26, 27, 28]. In this alternate model, the perception of sharp depth edges would be explained as illusions fostered by the presence of luminance boundaries much as occurs in color vision. The compelling percept of a *filled-in* surface is explained simply by the turning on of a low resolution symbolic flag indicating consistency with a smooth surface. An example of such an indication would be a high, sharp correlation peak in stereo matching. No explicit filled surface representation appears to be needed to explain psychophysical performance. This position and the idea of area-based measurements fits well with the measurement tool concept.

b The Sign-Correlation Algorithm

Binocular stereo, the measurement of optical flow, and many alignment tasks involve the measurement of local translation disparities between images. Marr and Poggio's zero-crossing theory [29] made an important contribution towards solving this disparity measurement problem. The zero-crossing theory, however, does not perform well in the presence of moderately large noise levels as has been illustrated by the inability of zero-crossing based approaches to solve transparent random-dot stereograms [26] – which are solved by the human visual system.

In addition, because of a different mindset, most implementations of the zero-crossing algorithm have been associated with cumbersome surface interpolation mechanisms which estimate likely

disparities between the sparse measurements obtained on zero-crossing contours. In reality no new information is added by the interpolation process [30]. Furthermore, it has been noted [30] that the intent of the original zero-crossing theory could be preserved while significantly increasing noise tolerance by doing area correlation on the sign of Laplacian of Gaussian ($\nabla^2 G$) filtered images. Rather than making pixel-wise measurements at sparse locations, we measure average disparities in specified regions. Explicit interpolation is no longer required, large noise levels can be tolerated, and most importantly, the broad sign correlation peak allows the direction (near/far) in disparity to the peak to be determined from dif-ferential measurements at a few locations. This greatly reduces the need for exhaustive search and allows accurate subpixel estimates of the peak position to be made.

The same zero-crossing information favored by Marr and Poggio is used by the sign-correlation approach, but the matching rule is changed from contour matching to area correlation. This subtle change makes a significant difference in the behavior of the matcher. Sign-correlation continues to provide useful disparity measurements in high noise situations long after the zero-crossing boundaries surrounding the signed regions cease to have any similarity. An intuitive explanation for why the two approaches perform so differently follows from the fact that the sign of the convolution signal is preserved near its peaks and valleys long after increasing noise has caused the zero contours to be fully scrambled. Thus area correlation of the sign representation yields significant correlation peaks even with signal-to-noise ratios of 1 to 1 as occurs with transparent random dot stereograms.

c PRISM-3 Implementation

Prior experience [30, 31] suggests that $\nabla^2 G$ operators with a center diameters, w, of at least 8 pixels – corresponding to a kernel diameter greater than 20 pixels – and correlation windows with diameters in the range of 32 to 64 pixels are desirable for image matching applications. Furthermore, these convolution computations must be carried out with the equivalent of at least 12 bits (and ideally 16 bits) of integer precision.

The sheer magnitude of the computation required to filter a pair of standard TV images at video rates with general 20 by 20 operators – some 6.2 billion multiply-adds per second – plus the additional overhead of computing correlations would tax a state-of-the-art supercomputer. However, by taking advantage of mathematical properties of the Gaussian convolution and making use of standard TTL technology, we have been able to implement such computations in the space of a few standard height VME bus boards.

We have just completed the design and construction of a third generation hardware accelerator for the sign-correlation algorithm. The PRISM-3 system's configuration is illustrated in Figure 2.

A nice property of algorithms tailored as measurement tools is that efficient hardware accelerators can often be designed to take advantage of their characteristics. In the case of PRISM-3, the sign-correlator is designed to make measurements at designated locations in the image. This makes it possible for a small device to make individual range or velocity measurements to subpixel resolution in approximately 200 microseconds.

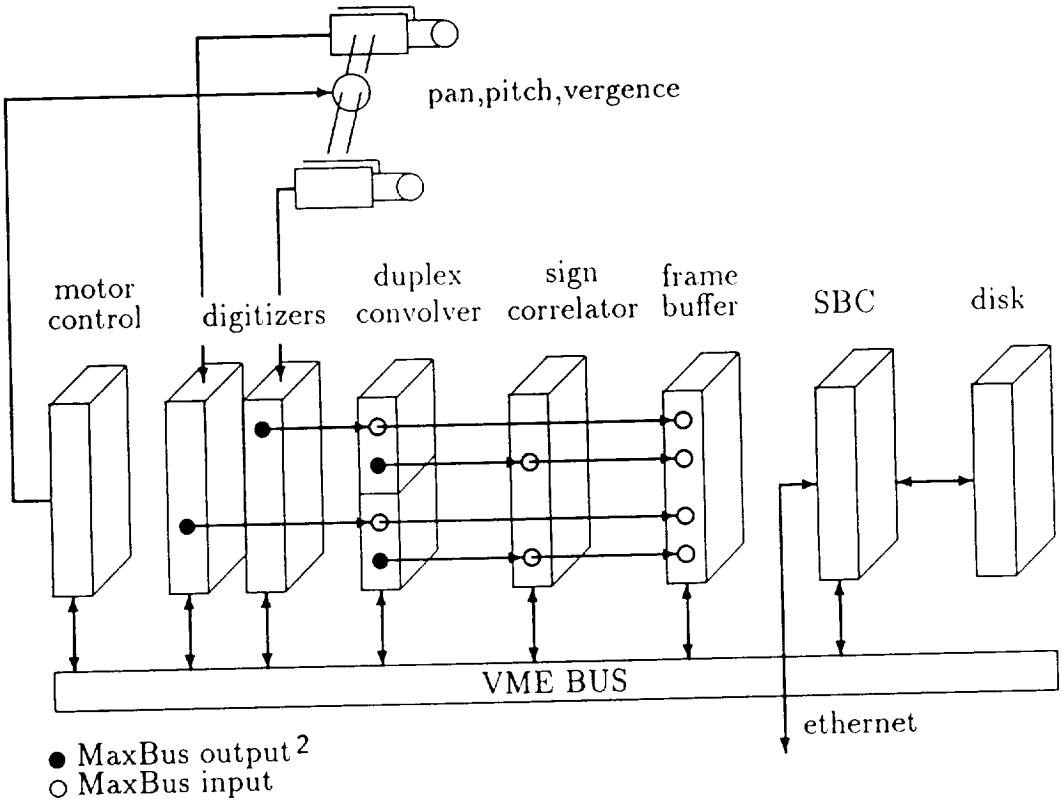

Figure 3.2: PRISM-3 System configuration

The complete system consists of an active stereo camera head and a desk top processor box containing all of the necessary processing hardware. The camera head has a pair of pixel-clock synchronized industrial cameras mounted on a head with pan, pitch, and vergence control. The binocular stereo video from the two cameras is digitized and filtered in parallel by video-rate Laplacian of Gaussian convolvers. Operator size is adjustable for a filter center diameter between 1 and 16 pixels in 1/2 pixel increments (the largest operator has a full extent of 43 by 43 pixels). 16 bit integer precision is maintained throughout the convolution pipelines. A specially designed binary correlator, capable of computing 36 separate one thousand point correlations in parallel in 100 microseconds, is then applied to make stereo or optical flow measurements off of the filtered images. A high-performance single board computer (SBC) controls the operation of the convolvers and the correlator and it evaluates mathematical formula for making subpixel disparity measurements from their results. The SBC also carries out low-level disparity to range conversions, controls the stereo camera head, and provides remote measurement services to external users via ethernet or VME bus transactions.

d Measurement Tools

The disparity measurements we make with the sign-correlation algorithm are averages over the correlation windows we use. This implies a limited spatial resolution, but we have found that this effect can be made to work for us in the design of a measurement tool that satisfies the three objectives set out in section 3.1.B.1

2. The MAXbus digital video interconnect standard was developed by and "MAXbus" is a trademark of DATACUBE, INC., Peabody, MA.

i Interpreting Peak Shape

The correlation peak shape obtained for a window containing a distribution of disparities is approximated by the convolution of that distribution with the single disparity correlation peak shape [32]. Thus, for example, when the window straddles a depth edge between two flat textured surfaces, the correlation function will have peak components contributed by both surfaces in proportion to their presences in the window. If the two surfaces are sufficiently separated in depth, two distinct peaks will be seen in the correlation surface. If the depth discontinuity is smaller, those peaks will merge to form a peak at the weighted average disparity.

We can take advantage of knowledge of how such average measurements behave as the correlation window is swept over, say, a sharp edge on a flat surface like a table top. This knowledge of the tool's behavior coupled with the user's knowledge of his visual world allow him to request measurement sequences that allow domain specific questions, such as where is the table edge, to be answered with high precision [33].

Other properties of the correlation peak are also potentially useful such as the width of the peak which may indicate the spread of disparities within the correlation window or the lack of matchable texture in the images. The peak height also carries some information about the noise level present.

ii Sign-Correlation Tool Design

The correlation peak shape appears to have the qualities we seek in a basic measurement tool – simplicity, understandability, and a rich behavior. Our intention is to provide users of the basic measurement system with access to the correlation peak shape so that discriminations like finding the precise position of the edge of a table top can be made.

We have yet to determine what the best way of reporting peak shape and position. Possibilities include:

1. Report raw correlation measurements. This is simple and general, but leaves the common tasks of computing peak position and width to the user.

2. Report peak height, width, and position. This does more work for the user, but makes it harder to discover less common but important events such as multiple peaks.

We plan initially to use the second approach but allow access to the full correlation data as well. As more experience is gained, we hope to arrive at a more refined output representation for the tool.

B.4 Next Steps

Our current research effort addresses questions in active vision concerning the intelligent use of measurement tools to accomplish tasks in dynamic environments.

a Figure-Ground Discrimination

One of the most basic problems in early vision is deciding what stuff goes together. Both binocular stereo disparity and optical flow field measurements are well suited for helping to separate out regions in the visual field that are coherent in range or motion. This is a form of figure-ground discrimination and can be effective even at a fairly low spatial resolution. Relevant tasks include:

1. identification of coherent regions in stereo disparity or motion;
2. determination of region characteristics such as mean position, size, direction of elongation; and
3. region boundary sharpening using other data such as intensity changes.

b Tracking and Following

Once we have a rudimentary set of figure-ground mechanisms operating, we will be able to focus on tasks involving active tracking of objects. Some of the tasks we plan to address include:

1. Stabilize image of moving figure on camera. This will involve investigation of motor control system design to minimize figure slippage during

   (a) steady motion,
   (b) accelerating motion,
   (c) figure in-plane rotation
   (d) figure rotation out-of-plane.

2. Follow figures undergoing occasional occlusion behind other objects. This task will build on basic stabilization routines, adding techniques for detecting the occurrence of an occlusion and making use of the figure's past dynamics to make estimates of where to look for it as time passes.

3. Make use of simple figure characteristics such as mean gray level or texture statistics, or color histogram to assist in following a figure such as a person in a crowded room.

4. Attempt to use change of vantage point to improve visibility of a figure being followed.

c Functional Search

One of the motivating themes for the measurement tool approach has been the concept of using those tools expertly to answer more complex task oriented questions. Examples of such questions are:

1. Determine that a specified volume of space is clear.

2. Ascertain the mean position, orientation and flatness of a patch of surface at a designated location.

3. Determine the position of a straight edge on a flat surface.

A few measurement routines like these, can be composed to accomplish more complex tasks such as finding a place to sit in a room where a place to sit is defined in terms of

(a) sufficiently large flat horizontal surface at proper height, with

(b) a vertical surface with sufficient area at one edge and above the horizontal surface,

(c) sufficient head clearance above, and

(d) foot clearance to the other side of the vertical surface.

There are likely flaws in this description, but one can imagine compiling a more sophisticated series of tests that make effective use of our measurement tools. A derivative of this endeavor

will be refinements of the measurement routines and the basic behaviors of the tools.

By studying demand driven tasks like these, we hope to learn how to make better measurements, and to learn how to integrate the use of a small set of basic measurement tools for accomplishing higher level tasks without compromising the theoretical integrity of the underlying computational theories for early vision.

C Task-Directed Visual Recognition

C.1 Introduction

The basic approach to visual recognition followed in this study involved representing images in the form of relational graphs, and matching these graphs to objects similarly represented. The relational graphs were automatically constructed over perceptually distinct image units, referred to as *blobs*. Given the relational graph representation, our strategy for object recognition was a voting-based search for partial matches of limited complexity.

The contributions made in this study include: a method of effectively extracting blobs of perceptually uniform and distinct color; a robust representation of the geometric and color arrangement of imaged object parts; an efficient method of identifying objects via relational match voting. An evaluation of the experimental performance of the algorithm applied to the recognition of real objects in real images is presented in Section 3.1.C.4; examples can be examined in Figures 6 and 7 of Section 3.1.D.

To participate in task-directed behaviors, a visual recognition module must respond to control that is a function of task and generate control information that can influence other aspects of the behavior. The recognition design developed in this study can participate in such behaviors; this is discussed in section 3.1.C.6.

C.2 Blob Extraction

The objective of the blob extraction process is to automatically represent, in compact form, the geometric arrangement and distribution of color among parts of an object's projection. Via a matching process, this information can be used to identify and locate objects represented in a similar fashion.

The basic strategy applied in the extraction process was to isolate regions in the image that are homogeneous and distinct in some property, and then estimate the rough position, extent, orientation and average color of each region.

a Finding Homogeneous Regions

There are two basic factors involved in determining a perceptually distinct and homogeneous region in the image: the geometry of the region, and the statistics of the visual measurements within and without the region.

Geometrically, a region is most visually apparent when simple, compact and reasonably continuous in representation over some area in the image. In this study, for reasons of speed in implementation, the geometry of a region was considered suitable if it was continuous; more specifically, a region was some single, connected component of the image. The general approach to region formation studied here was that of growing and merging region fragments; by constraining regions to connected components, the process of selecting fragments for combination was greatly simplified. The approximation of a perceptually distinct region as a connected component produced useful representations of the image data for the recognition

system discussed below.

Statistics of the visual measurements within and without the region are also important for determining the visual significance of a region. For this study, a region was considered visually significant if the difference between the average measurement in the region and the average measurement of each surrounding region was greater than some experimentally determined threshold.

Different region formation algorithms were developed and compared for quality and speed: (1) iterative merging, (2) one-pass region growing, and (3) two-pass grow and merge.

(1) Iterative merging. The process was initialized by defining a region for every sampled picture element, where the samples were measured at regular intervals in the image. The process then iteratively merged regions that were touching and mutually most similar in average measurement, until no touching regions had average measurement differences within the predetermined threshold.

(2) One-pass region growing. The image was assumed to be a 2D array of regular image samples. The process scanned the image once, and for every sample it performed the following operations. If the sample was similar enough to a neighboring region, it was merged; if the sample turned out to be a link connecting two regions that were similar enough, they were merged together; and if the sample was not similar to neighboring regions, a new region was formed initially containing only the sample. Region similarity was defined as in the first algorithm.

(3) Two-pass region grow and merge. This algorithm was a hybrid of the first two. The first pass of this process was the same as the one-pass algorithm. This was followed by one more pass where the regions of the previous pass were checked for neighbors that were similar enough, if so, they were merged.

Each of these algorithms were tested on several images, and the processing time and quality of the results were noted. The (1) iterative merging algorithm tended to produce the best regions (least fragmented), and the (2) one-pass algorithm produced the worst (most fragmented). This makes sense, since the first method does not converge until all possible merges of neighboring regions have been tested. However, the (3) two-pass region grow and merge algorithm tended to produce comparable results to the best one. This indicates that, for a high percentage of cases, most of a region's extent can be extracted in two merging passes.

In speed, the order was reversed. Algorithm (1) was generally five times slower than algorithm (2), while algorithm (3) was in between, but not much slower than (2). Because the quality of the (3) two-pass algorithm was close to the best, and its speed was close to the fastest, it was used in the final visual recognition experiments. Results of blob extraction using the two-pass region grow and merge can be examined in Figures 6 and 7 of Section 3.1.D.

b Color Representation

In this study, the blobs were extracted with respect to image color measurements: a region was considered homogeneous if its parts were similar enough in color and distinct if the neighboring regions were different enough in color.

In order to extract regions that correspond to visually distinct parts of scene objects, it is useful to base similarity on a color measurement that is principally a function of the object surface reflectance. Assuming that neighboring parts of the scene are receiving similar illumination color, such a similarity measurement can be achieved if color variation due to surface orientation

with respect to the light source can be factored out. Healey [34] has shown that, for near-white light sources, normalized red and green are principally functions of surface reflectance. Normalized red and green are defined as $r/(r + g + b)$ and $g/(r + g + b)$, respectively, for *r (red)*, *g (green)* and *b (blue)* standard sensor output. This is essentially true because, with white light, variation in surface orientation produces a variation in detected brightness, which is factored out by normalization.

Region extraction was attempted using similarity of normalized color. It was found that the regions fragmented and merged in places where the colors were subjectively similar and different, respectively. The performance on scenes with objects such as those in Figures 6 and 7 of Section 3.1-D seemed unsuitable for recognition. There are two problems with normalized color: spectral non-uniformity, and insensitivity to relatively unsaturated colors that are still easy to distinguish by their hues. Normalized color is essentially a projection of the usual three-component color onto a plane. This plane is parametrized by the normalized red and green components in a non-uniform fashion: distances between points near red or green are represented as larger than those between points near blue. This generates a non-uniform representation of distance with respect to hue, the perceived position of the color in the color spectrum. During region formation, red or green objects will tend to have more fragmented regions than blue objects.

The saturation of colors (how pure they are) also adversely affects the computed color differences in normalized color space. Colors that are relatively low in saturation, but are still distinct from each other due to large differences in hue, are represented as very close, much closer than two bright reds that look indistinguishable.

Because of the above problems with normalized color and the fact that hue was subjectively and experimentally the most important factor, similarity in hue was used to determine if regions were to be merged or not. Hue was computed using the formula presented in [35]. Hue is unstable for pixels with very low saturation and brightness. For this reason color samples low in these values were ignored during region extraction.

c Blob Geometry

Once blobs are extracted from the image in the form of regions of samples, their basic geometric properties are measured. In this study, the principal model of recognition was of matching representations of geometric arrangements of parts. To achieve a basic representation of the geometric arrangement of blobs, it is useful to estimate their rough positions, orientations and extents. More specifically, the geometric attributes estimated for a blob were the position of the center of mass, the orientation of the principal axis, the length and the width.

The principal axis of the region point set can be successfully defined as the normal direction to the eigenvector of the region scatter matrix that is associated with its smallest eigenvalue. This method is discussed elsewhere [36,37].

The length and width can also be derived from the eigenvalues of the scatter matrix. The length is proportional to the square root of the largest eigenvalue, divided by the number of samples (minus one), and the width is similarly related to the smallest eigenvalue. In this study, it was determined that the proportionality constant used depends on the nature of the sample distribution within the region. For a region with a roughly uniform sampling across its extent, the constant was estimated to be 4. This value was used in the experiments shown in Figures 6 and 7 of Section 3.1.D. In these figures, the blobs are depicted as ellipses.

Using this appraoch, the position, orientation and extent of blobs can be rapidly calculated. Given a set of image sample positions (x,y), the computation chiefly involves summing up the x,

y, x^2, $y^2$and xy terms, and finding the eigenvalues of a 2x2 matrix. The summation can be performed during the region merging process, and the matrix computation is handled in the 2x2 case simply by applying the quadratic formula.

C.3 Recognizing Objects Via Blob Representation

The general algorithm developed to recognize objects in visual data is presented in the next two subsections. The specializations required to apply this general algorithm to recognition from blob descriptions is covered in the subsections that follow them. The section closes with an evaluation of an implementation demonstrated on several real images.

a General Recognition Strategy

Our basic recognition strategy has three aspects: (1) the use of relational graph representations of images and objects; (2) recognition by partial graph matching; and (3) the use of a voting process to determine the best partial match.

(1) A relational graph of an image is composed of elements and arcs. The elements represent detectable units of image information, and the arcs between the elements represent geometric and other visually detectable relationships of the associated units. For this study, the units of image information are the blobs discussed above, and the relations are functions of their color and geometry. Relational graphs can be used to represent information salient for object recognition in images undergoing a high degree of variation due to changes in viewing conditions and object articulation [38, 39]. In this study, the images and the object models were both represented as relational graphs, and recognition was achieved by finding matches between these graphs.

(2) A complete match between the model and image is generally impossible in real images, and the search for the most complete match is a complex problem that is not realistic for fast recognition. Fortunately, it is also generally not necessary for confident recognition; usually only a fraction of the relations representing the image have to agree with those representing the modeled object to unambiguously identify the object's projection in the image. Our strategy was to recognize the object in the image by searching for partial matches of limited complexity.

Specifically, our strategy was to search for the best match of a single model element to a single image element. The best element-element match was defined to be the one with the most agreement between the modeled relations involving the model element and the image relations involving the image element, as represented in their respective graphs. This method effectively searches for the most complete matching subgraph that has a star topology: the single element considered the best match is the hub, and the supporting, matched relations with other elements are radiating arcs from this hub. In this study, the images and objects was modeled with completely connected relational graphs. Therefore, there was generally substantial support for each correct single element match in the form of a partial, star-graph match involving all of the detected elements of the model. In addition, we argue that this support can be efficiently computed.

(3) Our method of computing the support for a single element match was to accumulate votes for each single element match from each supporting (i.e., matching) relation, and then return the element-element match with the highest vote. Like other voting methods, such as geometric hashing [40] and generalized Hough transforms [41, 42], the complexity of the process was polynomial with respect to the number of image elements. In our case, the relational graphs were composed of unary and binary relations. The recognition process visits every pair of image elements, and the complexity was $O(n^2)$ for n image elements. However, unlike most other recognition algorithms based on voting, the object to be recognized did not have to be rigid, or

composed of elements that must be localized in the image with a high degree of reliability.

b Outline of Algorithm

The recognition algorithm proposed and experimented with had the following general structure:

1. Prior to recognition, model each object by building a relational graph representing its visual appearance.
2. During recognition, construct a relational graph representation of the image.
3. Next, for each modeled object, accumulate votes for the best (model element, image element) match:
   a. Initialize an array of vote accumulators, one cell for each potential (model element, image element) match.
   b. For each pair of model elements (m_1, m_2) and pair of image elements (i_1, i_2):
      i. Assess the match between the relations of (m_1, m_2) and the relations of (i_1, i_2). Assign a value from 0 (bad) to 1 (good).
      ii. If the assigned value is the highest for the triple (m_1, m_2, i_1), then add the value as a vote to the accumulator cell at (m_1, i_1).
      iii. If the assigned value is the highest for the triple (m_1, m_2, i_2), then add the value as a vote to the accumulator cell at (m_2, i_2).
   c. Return the (model element, image element) pair with the most votes.

The general matching algorithm was applied to the task of recognizing the projections of objects in real images represented as collections of blobs, as defined in Section 3.1.C.2. The details of applying the algorithm involved specifying the nature of the relational graph representation and assessing of goodness of match between model relations and image relations. These details are discussed in the next two sections, respectively, followed by a review of the results of this application.

c The Relational Graph Representation

In the first step of recognition, blobs were extracted from the image and a relational graph was automatically constructed over them. Prior to recognition, this process was also followed to automatically construct a model of the object from one of its views.

The relational graph of an image was composed of unary and binary relations, involving one and two blobs, respectively. A relation had two attributes: a type of measurement (e.g., hue, distance or angle) and a value for that measurement. For every image processed, the relational graphs constructed were complete in the sense that all blobs were assigned a unary relation of every type and all pairs of blobs were assigned binary relations of every type.

The overall objective in designing the relational measurements is the same as for any feature selection problem encountered when developing a recognition system: the feature should tend to

have a narrow distribution with respect to any one object and tend to differ in value from object to object. For relational measurements over the projections of three dimensional objects, this is achieved to a great extent by designing measurements that are relatively stable with viewing condition change and diverse in type. For this investigation, a color measurement invariant to surface orientation change and a relatively complete set of geometric measurements invariant to rotation, scaling and translation of the image (RST) were used.

A blob has color, position, length, width and orientation (when length and width are not equal). The aspect ratio (width over length) and hue of a blob were used as the unary relations. The aspect ratio is the only distinct geometric measurement of a single blob invariant to RST transformation in the image.

The blobs extracted were fairly saturated in color (Section 3.1.C.2). Assuming that the light source is roughly balanced in red, green and blue, this means that the hue of the extracted blob is relatively unaffected by changes in the object's surface orientation relative to the light source, and to the amount of the light source the object is receiving. Ideally, the saturation component of the color is also unaffected; however, due to the strong specular component of the object surfaces studied and the diffuse lighting of the scene, the perceived surface color saturation varied considerably.

The binary relations between two blobs expressed their relative orientation, size, and position (scaled to the size of the blobs). These measurements are also invariant to RST transformations [39]. The formulation of the actual measurements also took into account the instability of the blobs extracted. Sometimes one blob of the pair may have had reasonable extent estimates, sometimes the other, or sometimes just the lengths were reasonable and not the widths. To achieve some stability in the relative measurements, they were measured component-wise and redundantly, where possible. Relative size was represented as two components: relative width and length. The relative position of one blob with respect to another was represented in polar form and redundantly scaled by each blob, producing four different relative position measurements. More specifically, this approach resulted in the following seven binary relational measurements:

1. Relative orientation, measured as the angle between the blob principal axes (when defined).

2. Relative width, measured as the ratio of one width over the other.

3. Relative length, measured as the ratio of one length over the other.

4. Relative distance of the first blob to the second: the distance of the centroid of the first blob from the second blob's centroid, divided by the area of the second blob.

5. Relative distance of the second to the first: as above, with the blobs switched.

6. Relative angular position of the first blob to the second: the angular position of the first blob's centroid with respect to the axis of the second blob (when defined), about the second blob's centroid.

7. Relative angular position of the second blob to the first: as above, with the blobs switched.

Combined with the two unary measurements, nine different types of relational measurements were used, building a rich description of the geometric arrangement and color distribution of an image in terms of blobs.

d Computing Votes From Relational Information

During the matching process, a value is assigned to a match between model relations and image relations. This value is mapped between 0 (bad) and 1 (good), and is used as a weight for the vote of the match.

For each relational measurement, the value is a function of the difference between the model and image measurement. The mapping to the (0, 1) range should reflect the differences possible due to blob detection error, when the match is actually correct: if the average difference for the correct match is large, then large difference should be assigned similar values as small difference.

For the experiments discussed in the next section, the average differences in measurement were roughly estimated from 18 correct matches, and the following mapping was used from measurement difference to value: if the measurement difference d is less than the average when the match is correct a, then assign a value of 1; if $d < 2a$ assign a value of $2 - (d/a)$; and if $d > 2a$ assign a value of 0.

C.4 Evaluation of Method

The approach to recognition discussed here has been implemented and tested on multiple objects and images. The objects tested were commercial packages with multi-colored, complex designs. These designs were also covered with large print that often fragmented the extracted blobs. Figures 6 and 7 in Section 3.1.D show examples of the packages and matching results. At least four different packages were successfully modeled and recognized.

For at least two of the modeled objects, recognition was repeatedly tested against images from several novel views. In most of these test images, the colors represented in the models were not unique to the modeled objects, forcing the system to use both the color and geometric arrangement information stored in the relational graphs. In most cases, the two objects were correctly identified when visible in the scene. This included images in which the projection of the modeled objects were substantially occluded.

Since the demonstrated implementation of the recognition process always returned the best match, regardless of confidence level, spurious recognition occurred when the object was not visible in the scene. Though a confidence threshold was not estimated and applied at this time, these spurious recognitions were generally based on little support from the image data and were low in computed confidence. Thus a practical confidence threshold is feasible.

C.5 Color Invariants for Recognition

In the course of our investigations, some potentially useful color invariants were discovered that one could refer to as *normalized color ratios*. These values have not yet been studied experimentally, though they could lead to promising results.

Given the color imaging equations of Healey [34], the following functions of the three basic color components (red, green and blue) could be invariant to various changes in the viewing conditions. Suppose one has two color samples from two different points on an object's projection (r_1, g_1, b_1) and (r_2, g_2, b_2), compute the following ratios:

$$(r_1/r_2)/n \qquad (g_1/g_2)/n \qquad (b_1/b_2)/n$$

where n is the magnitude of $(r_1/r_2, g_1/g_2, b_1/b_2)$.

Additional invariants can be derived from the above three that may be more numerically stable:

$$(r_1 g_2)/(r_2 g_1) \qquad (g_1 b_2)/(g_2 b_1) \qquad (b_1 r_2)/(b_2 r_1)$$

Assuming that the two surface points are being illuminated by the same light sources, and that the sensor's spectral response can be approximated as three delta functions at some red, green and blue wavelengths, Healey's formulation predicts that these functions are invariant. Specifically, the surface orientations of the two sample points can be different, their orientation relative to the camera and light source can vary, and the illumination color can change.

The first assumption is reasonable under in most viewing conditions; however, it is uncertain how the sensor approximation of the second assumption affects the usefulness of the result. This could be analyzed and experimented with in future studies. Also, one does have control of the sensor response by use of filters. With appropriate selection of filters, the sensor response could be made sufficiently narrow-band.

C.6 Task-Directed Visual Recognition Summary and Conclusions

The basic approach to visual recognition followed in this study involved representing images in the form of relational graphs, and matching these graphs to objects similarly represented. Given the relational graph representation, our strategy for object recognition was a voting-based search for partial matches of limited complexity. The contributions made in this study include: a method of effectively extracting perceptually uniform and distinct regions of color (blobs); a robust representation of the geometric and color arrangement of imaged object parts; an efficient method of identifying objects via relational match voting. A visual process incorporating these contributions was implemented and applied to the recognition of real objects in real images.

To participate in task-directed behaviors, a visual recognition module must respond to control that is a function of task, and generate control information that can influence other aspects of the behavior. For recognition, the nature of the task can influence the location in the scene to be analyzed, the anticipated scale of the object's projection, the set of modeled objects to be searched for, and the specific image features to be emphasized. Given the design discussed above, our recognition process can be easily controlled in these ways by inputting images from specific camera orientations, by subsampling the image in meaningful ways, by pre-selecting the models of the objects of interest for matching and by weighting or ignoring the votes of certain selected features to reflect task-dependent biases.

In addition, valuable control information can be obtained from the recognition process in the form of the object identified, the location in the image or scene that the object may exist in, and the confidence in the match. In these ways, the recognition algorithm can be seen as a useful component of a task-directed vision system.

D A Layered Active Vision Architecture

We are developing a processing architecture and testbed scenario to implement and test our theories and ideas about interactive vision systems, particularly in the context of interacting with people. In this section we discuss the application domain, present the system architecture, and describe early experiments with the system.

D.1 Looking at People

Throughout the history of computer vision research, certain application domains have served to focus research directions and stimulate advances in the field. These domains have included the

"blocks world," factory and inspection tasks, and mobile robot vision. Although each of these areas has brought important issues into focus, each has had limitations. For example, the blocks world allowed researchers to move from low-level image operations to high-level recognition and constraint propagation. On the other hand, the blocks world assumes objects to have simple geometries, e.g., polyhedral. Inspection and assembly tasks require speed and robustness of processing, but often assume carefully structured environments and highly detectable features such as fiducial marks. Mobile robots require time-critical updates of spatial information, but are often limited to navigation tasks in relatively static environments.

In recent years, another application domain has begun to attract attention in the field: visual interaction with people, or "looking at people"[43, 44, 45, 46]. The surge of interest in this area has been due in part to widespread adoption of interactive computer interfaces, coupled with an optimism that machines are, or soon will be, fast enough to perform the necessary computations rapidly. Because of the variety of visual tasks involved in interacting with dynamic, non-rigid, articulated beings, this is a challenging application domain for the computer vision community.

Interaction between humans is greatly enhanced by the interpretation of non-verbal activities such as body movements and facial expressions. To achieve natural and convenient communication between humans and machines, computers must also be endowed with the capability to look at people and interpret visual cues for communication. This requires both interpreting the dynamic visual scene and understanding the conventions of "body language", i.e. the semantics of gestures, poses, and facial expressions.

Most of the research in computer vision devoted to interacting with people has concentrated on particular tasks such as face recognition, expression analysis, or body tracking. Although work in these areas dates back to the late 1960's [47, 48], there has been little effort to construct a theory of looking at people or a general framework for the various related tasks or behaviors.

In this research, we are exploring a framework for looking at people. The processing architecture we will describe is an initial attempt at constructing the building blocks necessary to achieve vision-based human-computer interaction. Because the tasks are inherently interactive, we pursue an active vision or "interactive-time" vision approach to the problem of looking at people. The term "interactive-time" is intended to suggest routines which are closely coupled with the reaction time of the process or agent calling the routine [49]. The reaction time constraints may range from on the order of milliseconds (e.g., feedback for controlling a robot's motion towards the human) to seconds (e.g., searching a room for people).

D.2 Organization of the Active Vision System

Enormous run-time resources seem to be required for such perceptual tasks. Early visual processing as conventionally conceived is computationally intensive, and because of noise and complexity at the signal level, it appears necessary to sift through massive amounts of data to recover stable, meaningful information. In biological systems, this is done in real time, using highly parallel, noise-tolerant techniques. Computer systems that attempt to do similar processing have tended to be slow and expensive.

The fact that massively parallel biological computations seem to do early processing of images in a more or less uniform way does not imply that engineered systems must do the same in order to support a given level of information extraction. We conjecture that much sparser early processing can be sufficient if intelligently applied, and that this process can be controlled with mechanisms similar to those emerging from recent research in intelligent reactive control.

In addition to the problem of computational resources, there is a large conceptual gap from signal to interpretation – from low-level to high-level data. Individual pixels contain very little

information, and the information they do contain is only understandable in the most basic physical terms. High-level descriptions of natural objects, on the other hand, are hard to model in purely physical terms, and therefore are hard to relate directly to low-level data. We use a hierarchical design to help bridge the gap from signal to qualitative description. Intermediate levels of description, such as blobs or regions, are taken to represent persistent visual entities. These data objects are updated from frame to frame in a way that reflects the incorporation of new measurements, as well as changes that reflect world dynamics.

a Overall Organization

The processing architecture is divided into three layers, loosely described as the "measurement" layer, the "blob" layer, and the "object" layer, as shown in Figure 3. Although there are similarities with the traditional layered computation vision approach (e.g., the layers are modular and somewhat independent), the architecture is more directly inspired by an active vision approach.

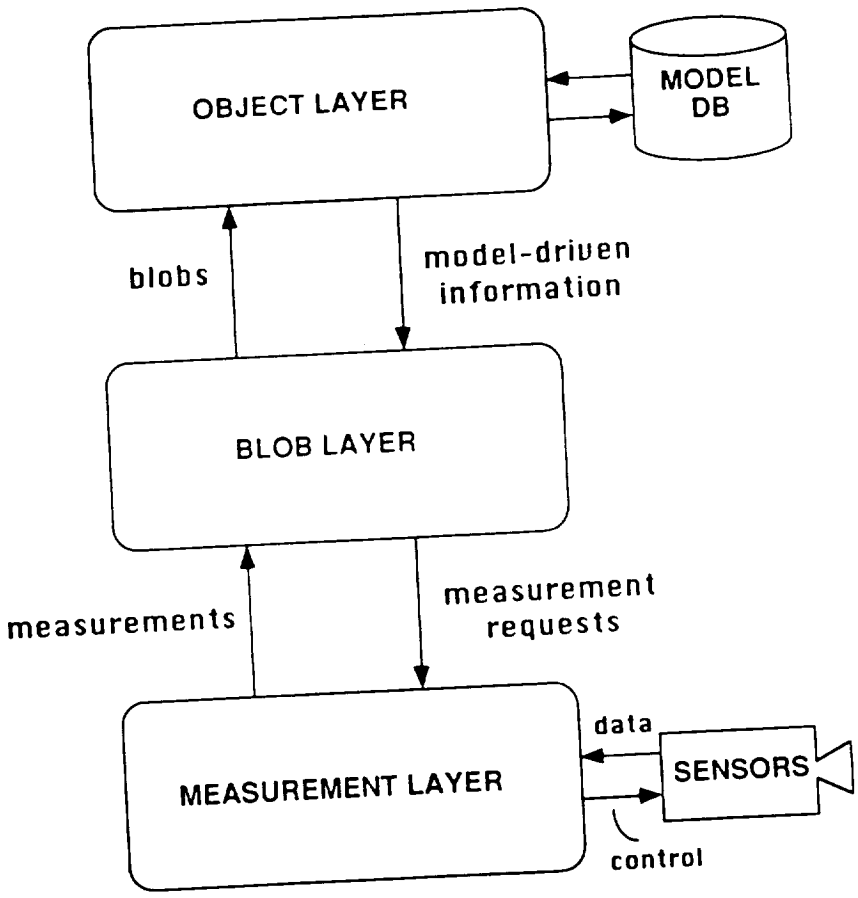

Figure 3.3: The processing architecture

The architecture exemplifies the following principles:

- The processing layers are modular and hierarchical. This helps bridge the gap from low to high level descriptions.

- The sensing is based on multiple modalities. This decreases the dependency on a particular modality or sensor type and increases the robustness of the system.

- The current state of each layer is available at regular, short intervals. This is consistent with the interactive-time requirement.

- Uncertainty estimates are associated with the information at each layer. At any time a module should have both a description of its state and confidences associated with the data structures (e.g., objects or blobs) which it is tracking. Directing attention toward or away from certain events or data serves to increase or decrease the confidences.

- The architecture supports both model- and data-driven processing. There is no a priori commitment to bottom-up or top-down strategies.

Each layer is constantly updated and may be queried at regular intervals. Although there are processes within each layer operating at various time scales, a fundamental property of the architecture is that at each "time click" the basic elements – measurement values, blobs, and objects – have meaningful values with associated error estimates. For example, if a moving blob is updated at time t1 and the object layer inquires about it at time t2, the blob layer may have to estimate its position based on its prior position and velocity, including a certainty measure which is a function of the measurement modality and blob-finding algorithm used, as well as the prior information.

b Measurement Layer

The measurement layer responds to requests for measurements, which may be in the form of pixel values, depth measurements, velocity vectors, color or texture measures, edges, or some other low-level measurement which can be quickly computed. The measurements may be from a single location – a "point" measurement – or the difference between two locations – a "comparison" measurement. The measurement syntax is shown in Table 1.

The AVP measurement layer is accessible by a uniform interface which hides the low-level sensor dependencies (e.g., control of PRISM and color digitizer). There are also built-in display and debugging tools for visualizing the output of the various modalities. Each measurement type has an expected time of retrieval, so that a process requesting a set of measurements can compute an estimate of the time it will take to fulfill its request. Also available are low-level behaviors such as tracking points in the scene and changing sensor parameters such as the sensor location and direction.

The suite of measurement types available at any given time is dependent on the available imaging hardware. However, the interface to the measurement layer assumes nothing about the hardware configuration, so in principal it will perform with any given supporting hardware – as long as the layer requesting the measurements can pursue its goals using varying measurement modalities. This implies an open architecture model, in that adding (or deleting) measurement types adds to (or subtracts from) its functionality. This concept encourages algorithms which rely on relative measurements that may be meaningful across several modalities. For example, a middle-level query "okay-to-merge?(blob A, blob B)" may be answered using range, color, and velocity information, or perhaps only color if range and velocity are not available.

Our implementation currently uses the PRISM-3 system and a color camera and digitizer as hardware resources to the measurement layer. The PRISM-3 system comprises a pair of stereo gray-scale cameras mounted on a pan/tilt mechanism, along with hardware and embedded software to support imaging operations and mechanical control. For example, the hardware performs very fast sign correlation to support multiple stereo depth measurements or motion vec-

```
(make-measurement type class location & key point scale)
```

where
type = { gray color texture depth motion }
class = { point comparison }
location = { XY-list List-of-XY-lists }
scale = { 1 3 5 7 ... }

Types

gray - Gray scale values averaged over a local gaussian patch
color - Color (R G B) values averaged over a local gaussian patch
depth - Range measurements via PRISM
motion - Image flow measurements via PRISM

Classes

point - Single (locally averaged) measurement
comparison - Difference of two point measurements

Location

XY-list - (x y) in image coordinates for point measurements
List-of-XY-lists - ((x1 y1) (x2 y2)) for comparison measurements

Scale

scale - NxN support for local gaussian averaging

Table 1. Measurement layer syntax.

tors per frame time. The color camera is mounted on the PRISM head and returns color or gray-scale images with a wider field of view. We are also beginning work on a workstation-only version which seeks to approximate the PRISM-3 measurements entirely in software.

c Blob Layer

The blob layer is responsible for segmenting and updating meaningful regions in the scene. Blobs are defined to be convex regions relatively uniform in some measurement type (or conjunction of types) – e.g., a blob may be a region of continuous depth or similar color. The blob layer is responsible for sending measurement requests to the measurement layer, and may query the object layer for hints on where to look for new blobs. The layer's processing is influenced both by the measurement layer (data-driven) and the object layer (model driven). There is in fact no a priori commitment to "bottom-up" or "top-down" processing, as the architecture can implement either or a combination of both. The layer provides processes for bootstrapping (finding blobs from no previous data) and tracking (updating known blobs), as well as mixed modes which combine updating known blobs and looking for new ones. The blob layer can be directed to look in certain areas and at a specified scale.

Algorithms used by the blob layer to detect and update blobs must return useful information in a short cycle time. They are required to make efficient use of limited resources (i.e., processing time) largely by using sparse measurements and/or measurements in limited regions of the visual

field. We are working on salient blob-finding techniques based on color, range, motion, texture, and combinations of these modalities.

We currently model blobs as ellipses. A number of other parametrized models may be useful, such as snakes or 2D superquadrics, although computational expense must be limited. These rather generic shape descriptions allow for either general or specific commitments to the data. A human, for example, can be modeled as one large blob or several neighboring and partially overlapping blobs. Keeping track of the relationships between articulated parts and the blobs which comprise them, at different scales, is the job of the object layer.

d Recognition Layer

The object layer seeks to fulfill requests from a user or planning agent. The queries may be as specific as "Where is the left arm?" or "How many people are there?", or as general as "What is of interest in the scene?" It instantiates stored object models and builds new models. It interacts with the blob layer by sending information about expected blobs, and by regularly querying the current state of scene blobs. In a completely bottom-up mode, the object layer interprets current and past blobs as best it can, given a database of known objects. In top-down mode, the layer tells the blob layer which blobs to track and where to look for new blobs.

Objects are modeled as collections of constrained volumetric parts, where the 2D projection of the parts can be approximated by the blob shape descriptions. The object model constrains the relative positions and orientations of its parts. An articulated object such as an arm is composed of three jointed parts – the upper arm, the forearm, and the hand – which are mutually constrained. The hand is further submodeled as several articulated parts. This simple model supports a hierarchical, coarse-to-fine search process, which may be terminated at the level appropriate to a given task. For example, "find the person" requires much less detail than "find where the hand is pointing."

Figure 4 shows an example of the interaction among the layers while looking for a person at rather coarse scale. The current state is shown on the left: the measurement layer returned a range and color description of the scene; the blob layer found some blobs related to a person and some other extraneous blobs; and the object model being sought approximates the human component parts and their (2D) geometric constraints. The right side of the figure depicts subsequent processing which seeks to update the description: the blobs are matched as well as possible with the object model, and missing model parts are noted; the object layer tells the blob layer to track some of the blobs, and to search for new blobs in the area of the missing arm; the blob layer turns that request into a measurement request – in this case a request for an array of range measurements in the region of interest.

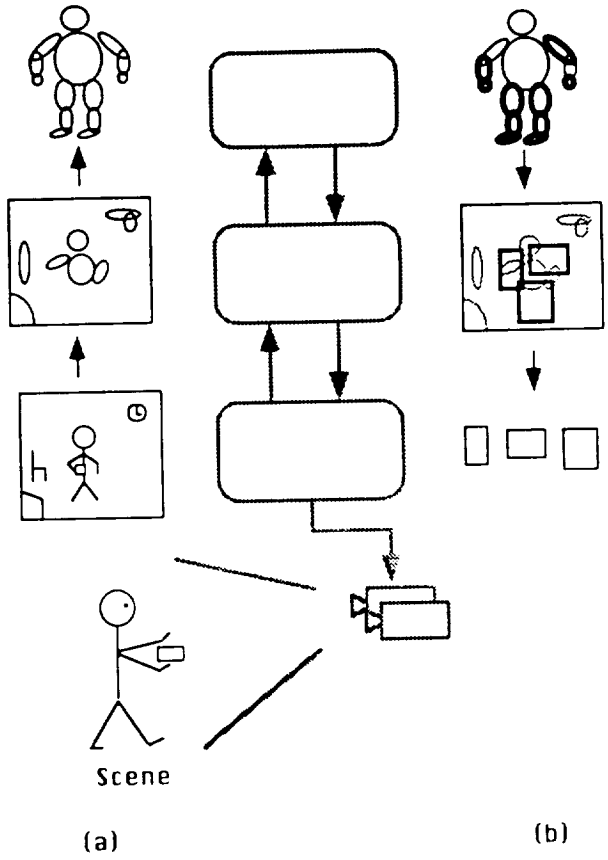

Figure 3.4: Layer interaction.
A flow of information is shown from the measurement layer up to the object layer (left side), and then back down to the measurement layer (right side). (Note that the layers do not wait for the completion of such a cycle, but are constantly active servicing their most recent requests.)

D.3 Experiments

We are developing a testbed scenario in order to facilitate development of the system and to gain experience in a realistic environment. The scenario is a package-handling situation, where a "customer" interacts with a robot behind the counter. In our laboratory, a gantry robot is set up to move in an approximately 4' by 8' workspace. (See Figure 5.) It has a 5 DOF arm and the PRISM-3 active camera head installed on top. On one end of the workspace is a collection of boxes and objects (the storeroom). On the other end there is a counter. People can walk up to the counter and request services from the robot. Human actions include:

- Requesting the robot's attention
- Handing a package to the robot
- Taking a package from the robot
- Communicating via simple gestures (e.g., nodding or shaking one's head)

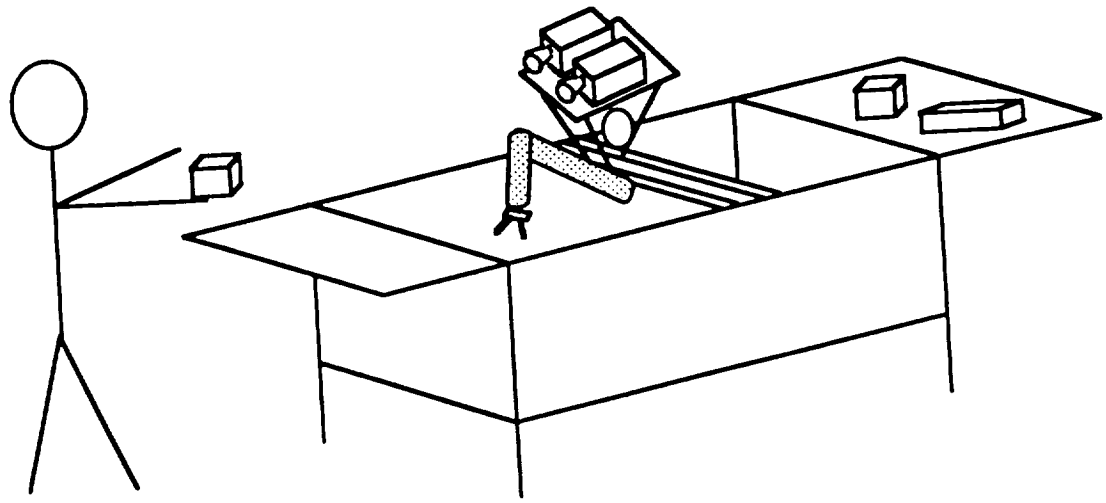

Figure 3.5: Testbed scenario.

To successfully react to these events, the robot must support a significant level of interaction. It must be able to:

1. Notice and locate the presence of a person
2. Initiate a dialog to discern the "customer's" desire
3. Understand simple gestures, e.g., to discern "yes/no"
4. Recognize the "package offering" posture
5. Recognize the "package receiving" posture
6. Distinguish one customer from another

Currently some of the behaviors streamline the general layered architecture by essentially operating closed-loop on the measurement data. For example there is a "find the head" behavior which, if expecting a person in the customer area, used a simple depth search routine to locate, and subsequently track, the customer's head. The behavior is reliable when one person is in the customer area. A related direct behavior is discerning a "yes/no" answer by looking for a head nod or shake. Once the head is located, its center can be tracked at frame rate by the PRISM system, and the path is easily classified as largely vertical – a "yes" – or largely horizontal – a "no." Our research objective is to develop algorithms that close the loop at higher and higher levels of the architecture while maintaining interactive-time response.

One experiment we are conducting along these lines is aimed at constructing object descriptions in a manner which is compatible with the layered architecture. We employ the voting-based recognition algorithm, described in Section 3.1.C.3, which manipulates graph representations describing relationships among blobs. These representations are used to model and recognize packages in the robot's workspace.

The recognition process involves two main parts: learning models and recognizing instances of the models. To learn the model, we isolate the package in the camera's field of view, as in Figure 6(a). The extracted blobs shown in Figure 6(b) are used to construct the object model. Learning objects is done off-line.

The recognition process seeks to instantiate the stored model in the cluttered scene shown in Figure 6(c). Figure 6(d) depicts the blobs computed from the scene, and the recognized package

which is marked and identified as object #0. (The blob shown is the blob with the highest support from the voting scheme described in Section 3.1.C.3.)

Figure 7 shows a similar experiment, learning a second object model and subsequently finding both known packages in the scene.

The blobs used in this process are constructed using a segmentation algorithm based on hue. Hue has proved to be a salient cue in this domain. For example, hair and flesh colors can be used to detect some person-related blobs, and colored clothing can be useful both in detection and in re-identification – for example, after the robot returns from acquiring a package it should quickly re-identify the person at the counter. Despite the utility of hue as a cue in this domain, we regard the current algorithm, based on a single modality, as merely a first step toward developing robust multi-modal segmentation.

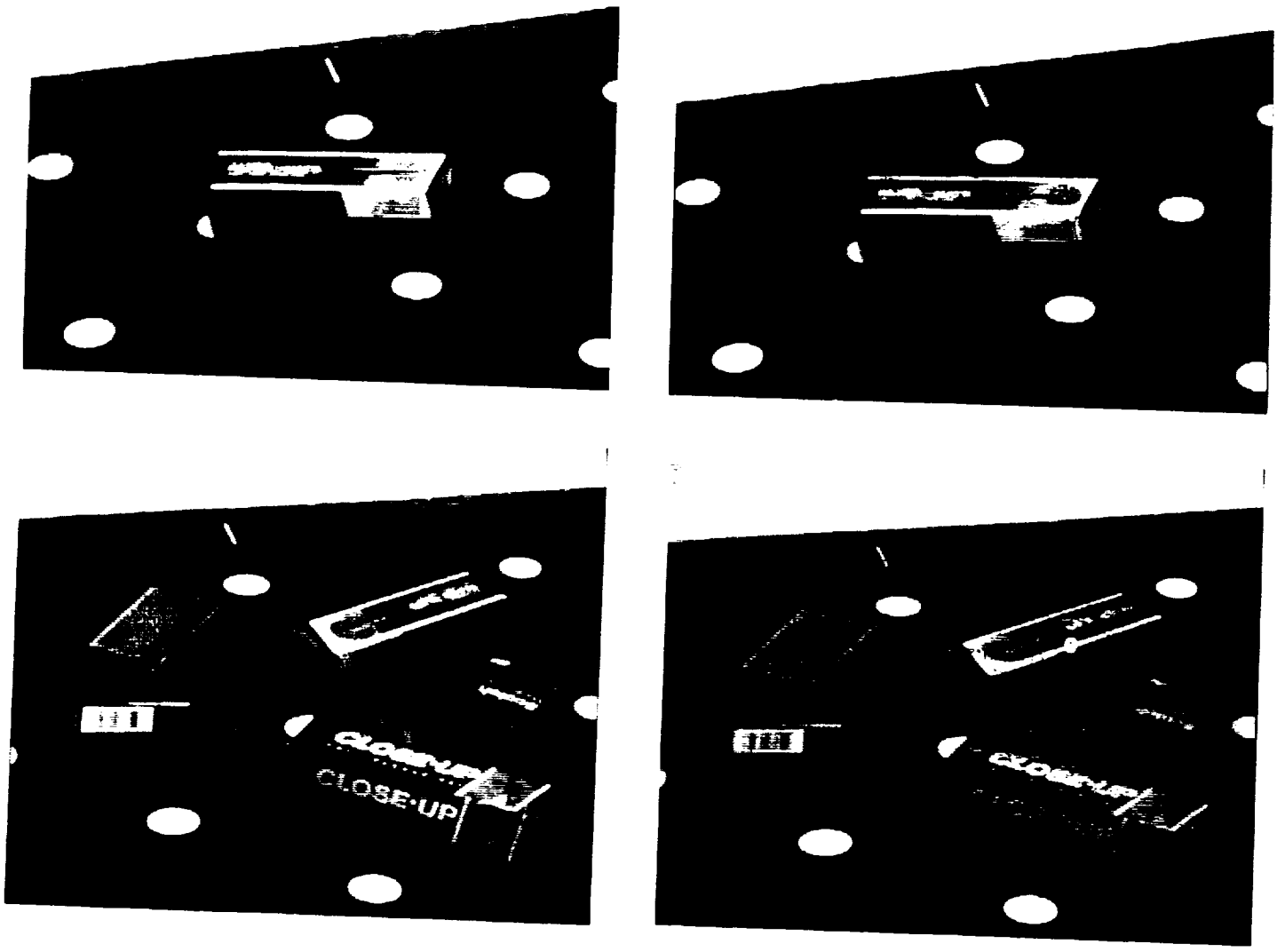

Figure 6: Package recognition example #1.

(a) Object to be learned.
(b) Blobs comprising the object model.
(c) Cluttered package scene.
(d) Blobs found in the scene; recognized object is marked and identified by object number.

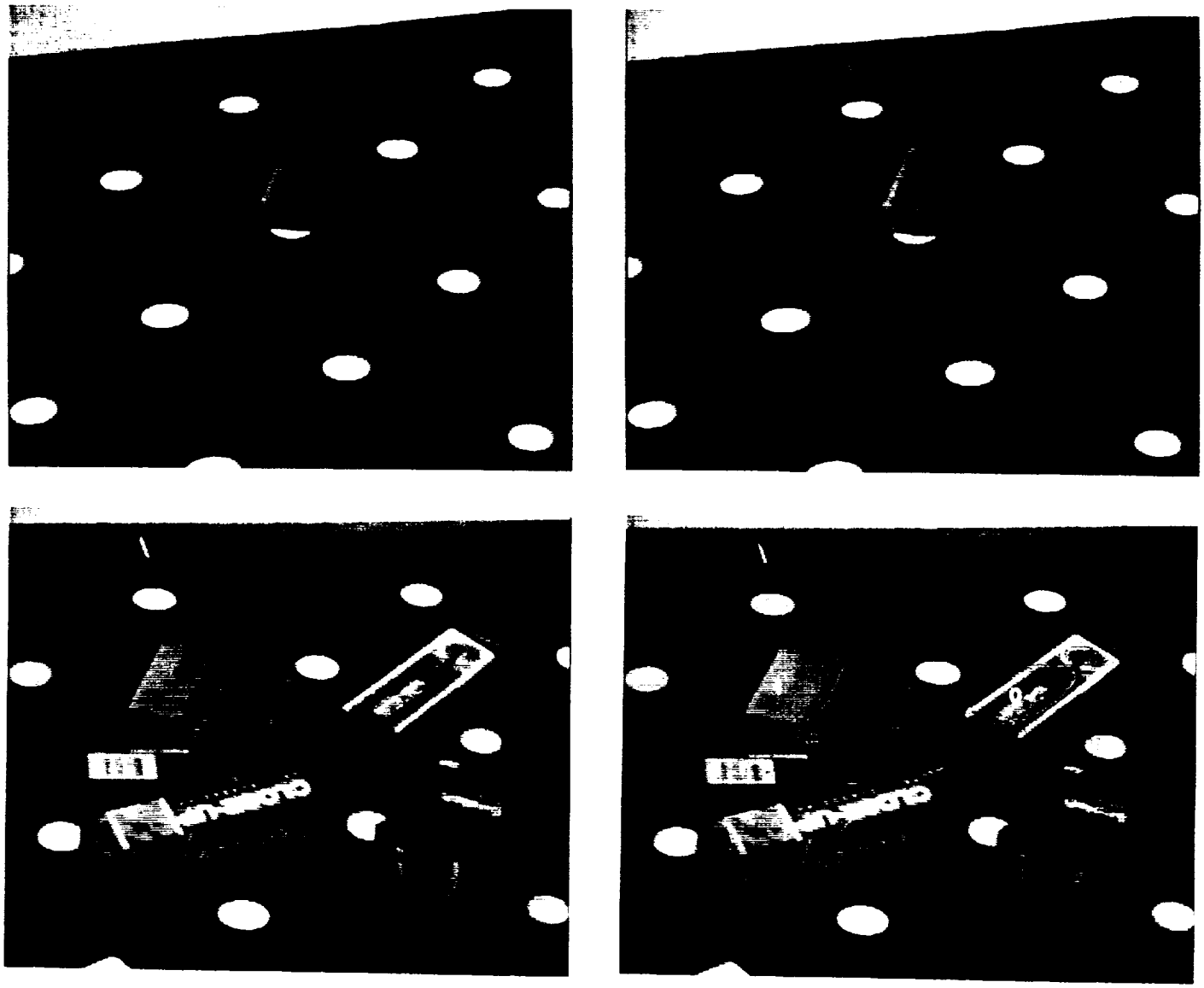

Figure 7: Package recognition example #2.

(a) Second object to be learned.
(b) Blobs comprising the object model.
(c) Cluttered package scene.
(d) Blobs found in the scene; recognized objects are marked and identified by object number.

3.2 Gaining Information Through Learning

Delayed reinforcement learning is an attractive framework for the unsupervised learning of action policies for autonomous agents. Some existing delayed reinforcement learning techniques have shown promise in simple domains. However, a number of hurdles must be passed before they are applicable to realistic problems. This paper describes one such difficulty, the *input generalization problem* (whereby the system must generalize to produce similar actions in similar situations) and an implemented solution, the *G algorithm*. This algorithm is based on recursive splitting of the state space based on statistical measures of differences in reinforcements received. The G algorithm's sound statistical basis makes it easy to predict when it should and should not work, which is not true of the sole previous approach to the problem.

A Background

Delayed reinforcement learning is a framework for learning, without supervision, to act in an environment [50]. In this framework, an agent is given on each tick, in addition to "perceptual" inputs, a numerical *reinforcement*, which is a measure of the *immediate* value of the state corresponding to the inputs. The goal of the agent is to choose actions to maximize the sum of reinforcement over time.[3]

Delayed reinforcement learning is an attractive framework due to its similarity to the problem faced by a person or other creature placed in unfamiliar surroundings and expected to act intelligently. Such problems are of increasing theoretical and practical interest due to recent progress in the construction of autonomous agents such as mobile robots. Though such systems have achieved new levels of performance, they generally depend on elaborate hand-coded policies in computing how to act. When the environment for which they were designed is perturbed, they may fail gracelessly; they are unable to adapt to new environments; and the process of hand-coding policies they require is slow and error-prone. Agents whose action policies are developed autonomously from a simple reinforcement signal might transcend all these problems [51].

The work described in this paper began with the goal of applying existing delayed reinforcement learning techniques to the problem of learning visual routines; that is, of learning to control an active vision system in which the sorts of visual computations performed can be controlled top-down. Our recent use of such visual systems [52] has depended, again, on explicit hand-coded action policies. Experiments combining reinforcement learning with visual routines were proposed in Chapman's thesis [51], and some simple visual routines have been learned successfully by a delayed reinforcement learning system of Whitehead and Ballard's [53]. We immediately ran into a series of technical problems, however, which made it clear that the existing learning techniques are not up to the task. This paper reports progress on some of these technical subproblems.

A.1 Temporal Difference Learning

The best-understood approach to delayed reinforcement learning is *temporal difference (TD) learning*, studied in particular by Sutton and his colleagues [54, 55, 50]. Two forms of TD learning have been studied in detail, the *adaptive heuristic critic* of Sutton [50] and *Q-learning*, due to Watkins [56]. Several authors have compared these methods empirically and found Q-learning superior [57, 58, 59], so we adopted Q-learning as our starting point.

Q-learning is based on estimating the values of $Q(i,a)$, which is the *expected future discounted reinforcement* for taking action a in input state i and continuing with the optimal policy. The discounted reinforcement is the sum of all future reinforcement weighted by how close they are;

3. Or, more accurately, to maximize a future-discounted sum of reinforcement, as we will explain.

specifically

$$Q(i(t),a(t)) = \sum_{j=0}^{\infty} \gamma^j r(t+j),$$

where t is the present time, $0 < \gamma < 1$ is a *discount factor* close to one, and $r(t)$ is the reinforcement received at time t. Thus the Q values say how good an action is not only in terms of immediate reinforcement, but also in terms of whether they get the agent closer to future reinforcement, and in effect allow the learner to dynamically create subgoals based on reinforcement that may be far in the future. Given Q values, we can also compute the value of a state as the value of the best action in that state:

$$U(i) = max_a Q(i,a).$$

The learning algorithm works by creating a two-dimensional table of Q values indexed by actions and inputs and then adjusting the values in the table based on actions taken and reinforcement received. This process is based on the observation that Q of the current (input,action) pair can be computed based on the immediate reinforcement received and the value of the next input:

$$Q(i(t),a(t)) = r(t) + \gamma U(i(t+1)).$$

For further details, see [57, 56].

A.2 The Input Generalization Problem

The first problem we faced in applying the Q algorithm to learning visual routines was that of too many inputs. Our target visual system [52] provides more than a hundred bits of input, corresponding to more than 2^{100} distinct inputs. This is a problem for two reasons: space and time. One simply cannot allocate an array one of whose dimensions is 2^{100}. Even if you could, it would divide the state space up into tiny pieces, each of which would occur extremely rarely, so the system could never accumulate enough experience to gauge the value of most states. Somehow a learning algorithm must guess about the value of states based on experience with similar states. But how can you know which states are similar when you have no experience with them?

The bulk of this paper describes an algorithm, the G algorithm, that addresses this generalization problem.

This problem has not previously been raised as such in the delayed reinforcement literature. However, an existing extension to TD does address the problem. Various researchers have combined TD with connectionist backpropagation, which can generalize given a large number of input bits [60, 58]. Some have had considerable success with this method, notably Lin [58]. Others have found the results disappointing [50, 57, 61]. Our own experience has been negative: in the domains we tested it in, the combination of TD and backpropagation learned very slowly

and converged to poor action choices. It is hard to set about explaining the discrepancy because backpropagation is ill-characterized. It is impossible to know how the algorithm will perform when presented with a complex problem; it often converges to bad local minima.

The G algorithm is a more direct approach to the generalization problem. It is mathematically well-characterized due to a sound statistical basis, and it is therefore easier to determine when and why it should or should not work.

A.3 The Domain

The domain used in this research was the videogame Amazon, previously used in Sonja [51]. More accurately, we studied one apparently simple subproblem from this domain, which turned out to be unexpectedly difficult.

In Amazon, the player controls an "amazon" icon which is attacked by ghosts. On each tick, the player can move the amazon one pixel in any of the eight "king's move" directions. Since the amazon is fifty pixels high, this means that the player has very fine-grained, effectively continuous, control over motion. (The otherwise similar videogame domains used by other researchers have coarse motion control, in which the player icon moves its entire width in a single tick. We'll see that this difference matters.) Ghosts similarly move a single pixel on each tick. The player can cause the amazon to shoot a "shuriken" in the amazon's direction of motion. Shuriken move in a straight line, four pixels per tick; if they collide with a ghost, the ghost dies.

The subproblem from Amazon we studied generates a ghost at a random distance and orientation from the amazon, waits for the amazon to kill it, and then repeats.

From earlier experience with Sonja, we knew of an encoding of this problem that is sufficient for efficient action. It is sufficient to know the direction from the amazon to the ghost and whether the amazon and the ghost are aligned along one of the axes on which shuriken travel. Then an efficient strategy is: if aligned, shoot in the direction of the ghost; otherwise move in a direction that will align you with the ghost.

We had to add an additional input for the sake of learning. It enables temporal backpropagation of the reinforcement for killing a ghost. Dozens of ticks can pass between the time the amazon shoots a shuriken and the time it reaches its target; somehow the system has to connect these two events. This is extremely difficult without additional inputs.[4] We added an additional bit of input which says whether or not there is a shuriken in flight that will intersect with the ghost if the ghost does not get out of the way. This bit effectively bridges the action of shooting on the right trajectory with killing the ghost.

This encoding of the scene is *not* what would be produced by a realistic visual system, and we intend to replace it with a more realistic encoding in future work.

The motor outputs from the learning system are three bits encoding the eight possible directions of motion and one bit that says whether or not to shoot.

The reinforcement given is 10 when a ghost dies, otherwise -.1 if a shot is taken, otherwise 0.

This apparently simple domain is difficult for several reasons. First, some states are very rare, and it is hard to gain enough experience with them to find an optimal policy. Second, the ghosts'

4. It's impossible, in fact with TD methods in which λ, the TD recency parameter [51], is zero. We did not experiment with $\lambda>0$ but we expect that learning would converge very slowly if at all.

behavior is so programmed that they "try" to stay unaligned as long as possible. When the amazon and the ghost are unaligned, there are limited opportunities for the system to learn anything interesting. In particular, the system is not (locally) reinforced for actions that lead to killing the ghost, because it is impossible to kill the ghost while unaligned. Under a random strategy, the amazon and ghost will typically stay unaligned for stretches of several hundred ticks punctuated by brief periods of alignment terminated by ghost death. Thus most experience is of very limited value, and the interesting reinforcements (ghost deaths) occur infrequently. Third, there is a great variance in the value of states that the learner cannot perceive. How nearly aligned the ghost is and how close it is to the amazon determine how long it will be before it is possible to kill it. If the system gets a series of "easy" ghosts in a row it can readily come to wrong conclu-

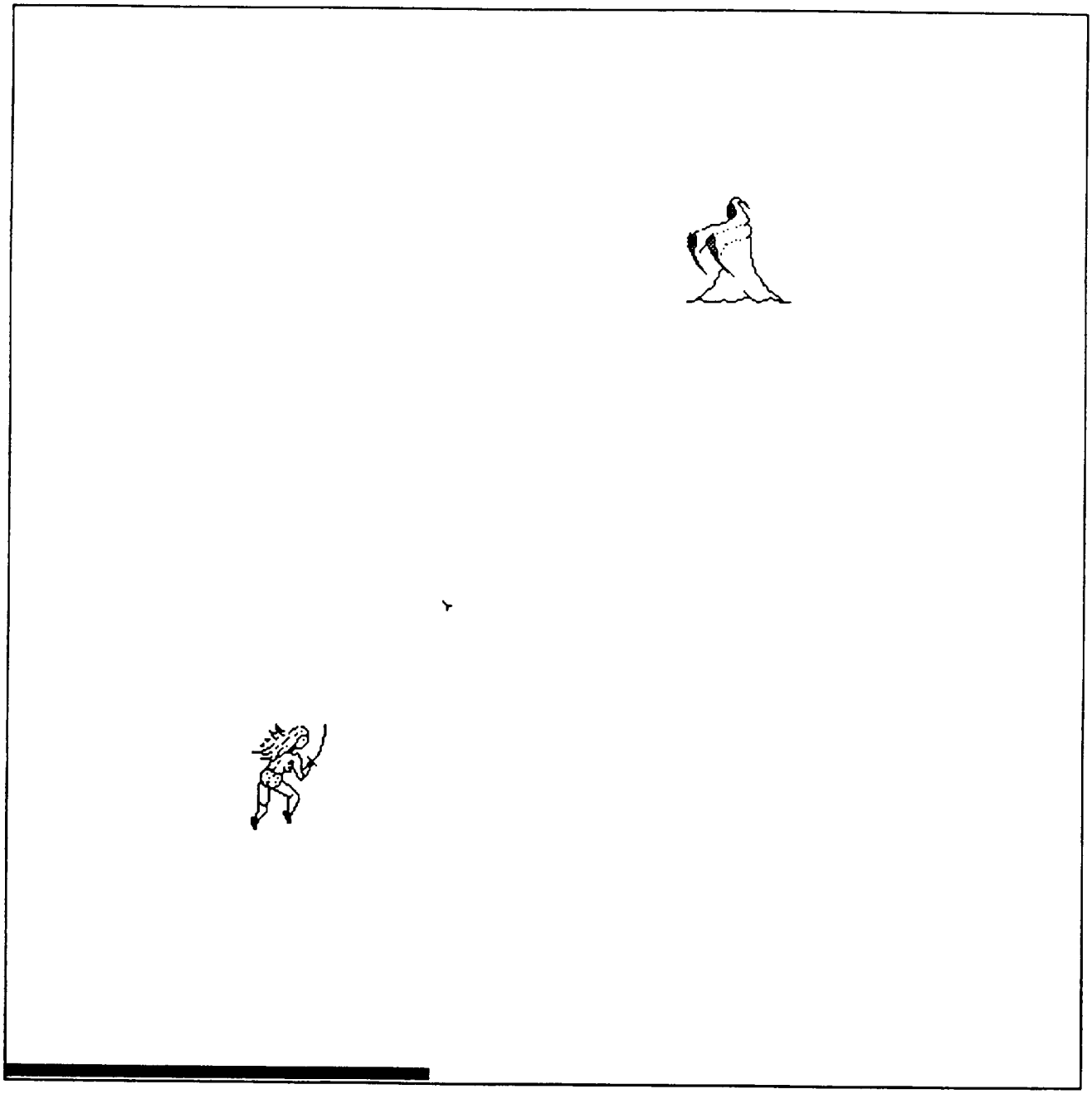

Figure 3.8: The amazon has shot a shuriken at the ghost.

sions about a state value. This just means that the learning rate must be set sufficiently low to even such differences out; but that in turn means that learning is slow. Finally, there is a more general problem of inadequate perceptual access. The problem seems easy to a human player who can see what is happening on the screen, but the same problem when reduced to a few bits of input becomes very difficult. These last two difficulties are artifacts of the domain encoding and we intend to eliminate them in future work.

B The G Algorithm

We can motivate the G algorithm in two ways. The first way is to see it as collapsing the exponential sized Q table engendered by large numbers of input bits. In most domains, large chunks of the table should have identical entries, because many of the bits will be irrelevant to acting under certain circumstances. If we can figure out which bits are irrelevant, we can summarize a large region of the state space with only one Q value, thereby saving both space (to store the values) and time (since experience with any state in the region can be used to update the single Q value).

Another way of looking at the algorithm is as a technique for incrementally constructing the sort of action selection networks that have recently been used by various researchers (including ourselves) in agents interacting with environments [62, 63, 51, 64, 65, 66]. These networks are digital circuits that compute what to do based on perceptual inputs. The circuits are kept shallow (with short paths between inputs and outputs) in order to compute quickly when implemented on slow, massively parallel hardware. Typically they maintain little or no state. Because the networks are shallow, they have the property that although every input bit is used in computing how to act, in most situations most bits are ignored. For example, whenever Sonja detects that it has lost track of the amazon, it searches for the amazon; all other inputs are irrelevant in this case. Thus a single input bit can determine how to act in some cases.

The G algorithm incrementally builds a tree-structured Q table. It begins by supposing that all input bits are irrelevant, thereby collapsing the entire table into a single block. It collects Q values within this block. G also collects statistical evidence for the relevance of individual bits. When it discovers that a bit is relevant, it splits the state space into the two subspaces corresponding to the relevant bit being on and off. Then it collects statistics (action and relevance) within each of those blocks. These blocks can in turn be split, giving rise to a tree-structured Q table. The system acts on the basis of the Q statistics in the leaf node corresponding to an input. Thus the tree acts as a boolean input classification network, essentially similar to the sorts of action networks described above.

This incremental, one-bit-at-a-time construction of the G tree puts a constraint on the sorts of environments that G can learn in: the relevance of bits must be apparent in isolation. The algorithm will fail if groups of bits are collectively relevant but individually irrelevant. If we consider the perceptual system of an agent to be part of the "environment" of its learning system, as we must, then this constraint can be placed on that system rather than the world. In other words, we hypothesize that *a well-designed perceptual system orthogonalizes inputs such that they are individually relevant.*

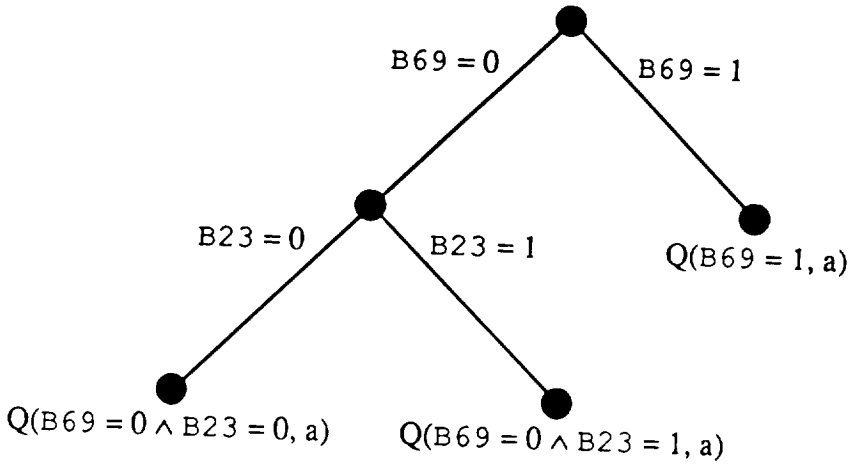

Figure 3.9: A G tree.
Internal nodes correspond to input bit splits; the algorithm collects statistics about the subspaces represented by the leaf nodes.

C Statistics

This section describes the statistics used in G to decide when to split and how to act.

C.1 Bit Relevance Tests

G uses a standard statistical test, the *Student's t test* [67], to determine when a bit is relevant. The *t* test tells you, given two sets of data, how probable it is that distinct distributions gave rise to them. That is, how likely is it that these two sets of data arose from the same underlying process? This is just what we need in order to determine whether an input bit is relevant: is the learner/environment interaction the same when the bit is on versus off, or is it different?

Two sorts of relevance statistics are kept: a bit may be relevant because it affects the value of a state or because it affects the way the system should act in that state. Two sorts of statistics are used to determine value, corresponding to the mean *immediate* value of the state and its mean *discounted future* value. Both sorts are required: immediate value is used to "bootstrap" the process by recognizing the states that themselves give large reinforcements (those in which a shuriken is flying toward the ghost, in Amazon) and discounted value is used to find states that lie on a path toward externally reinforced states (such as those in which the ghost and the amazon are aligned). For each bit in each state block, G keeps track of the immediate and discounted values of the state block subdivided by the bit being on and off, and compares these values with the *t* test.

A bit may also be relevant because it affects how the agent should act; for example the four direction input bits in Amazon do not affect the values of the state, but they do determine which direction the player should head in. To discover such relevance, G keeps track, for each action in each state block, of the discounted value of taking the action in that state block when the bit is on versus when it is off, and compares these values with the *t* test.

When a bit is shown to be relevant in a block, that block is split on the bit. When a block is split, all discounted statistics (both action value and relevance) must be zeroed. The reason is that a state block whose mean value is low may have a subblock whose value is high. Before the split

is made, this high-valued subblock is effectively invisible, and the estimated values of all states that can transition to that subblock will be too low.

Throwing away all experience accumulated thus far on each split seems too drastic. We are exploring ways of doing this, and expect that they will substantially increase the learning rate.

C.2 Exploration and the Interval Estimation Algorithm

Any delayed reinforcement learning system faces a fundamental problem: the tradeoff between acting to gain information and acting to gain reinforcement. At any given time, the system has an imperfect approximation to the optimal policy for acting in its environment. In order to improve the policy, it must often take actions that appear suboptimal, because its information about those actions may be incorrect. Most systems in the literature address this problem by occasionally choosing random actions, typically with probability decreasing over time and inversely related to the difference between the apparent value of the action and the apparent value of the optimal action.

We took a different approach, using Kaelbling's *interval estimation* (IE) algorithm [57], which has a sound statistical basis. This algorithm uses information about the range of reinforcements received for particular actions to construct a confidence interval for the mean reinforcement. The system then chooses the action whose reinforcement interval has the greatest upper bound. A high upper bound can be due either to a high mean or to a high uncertainty about the value of the action. This algorithm has been shown empirically to be superior in practice to available alternatives [57].

C.3 Enforcing Normality

The IE exploration strategy and the t test are only as good as the statistical model they use. Unfortunately, IE and the t test depend on assumptions about the statistical distribution of the items sampled. In particular, they depend on the assumption that the approximations to the discounted future value of actions in states are distributed normally. Most statistical techniques make such assumptions of normality.[5] The normality assumption, while approximately accurate in the simple domains we studied, is violated by Amazon, because the interesting reinforcement value (for killing a ghost) occurs so rarely. We found, as a result, that the G algorithm as specified in the last section frequently made incorrect choices both about action and relevance.

Normal statistics are frequently used to examine non-normal data, and this is often successful due to the *central limit theorem* which states that the sum of a set of values from an arbitrary distribution will approach normality as the number of samples increases. We were able to solve the problems described in the last section by delaying decisions based on statistical information until enough samples had been collected for them to approach normality.

Bad IE action value statistics can cause the system to converge to incorrect Q values. The Q algorithm provably converges to correct Q values so long as every action is executed infinitely many times in every state (and various other conditions are satisfied) [56]. However, the IE strategy is not guaranteed to execute every action infinitely many times if it gets a sufficiently bad early sample.[6] This problem can be overcome by postponing applying IE to a situation and acting at random until all actions are tried at least k times [57].

5. The alternative is to use nonparametric statistics, which are unwieldy and seemed inappropriate to this domain (for reasons too complex to go into here).

6. Some other exploration strategies in the literature do probabilistically guarantee infinite execution, but are *ad hoc* rather than statistically motivated, and may not execute exploratory actions sufficiently often to eliminate this problem in practice.

Similarly, insufficiently many samples for the t test can make bits look relevant that are not. We thus required a certain number of samples before using the test to split a subspace.

These fixes depend on numerical thresholds whose values may vary according to the domain, and are thus not satisfying. A better-motivated alternative would be to use statistical tests of normality (such as skew and kurtosis [68]) to decide whether enough samples have been collected to trust the data. We have not implemented such tests.

C.4 Low Frequency Noise

The techniques described in the previous section were mostly sufficient in practice to ensure that the system did not split on irrelevant bits. An additional problem arose in some cases, however: while bits that changed rapidly presented no problems, irrelevant slowly-changing bits continued to pass the t test. Figure 10 illustrates the reason. If an input bit changes slowly relative to changes in estimated state values, the statistics collected to determine the discounted value of a subspace are skewed. In the figure, the estimated value of the state starts low and converges to a higher value. Initially the bit B69 is low, and later goes high. As a result, it will appear that *B69* being on makes this subspace more valuable and the system will split.

The solution to this problem is to separate learning into action value and bit relevance phases. Estimated Q values are held constant while bit relevance statistics are collected. The system switches phases when values seem to have settled down, based on information about the derivatives of the statistical measures.

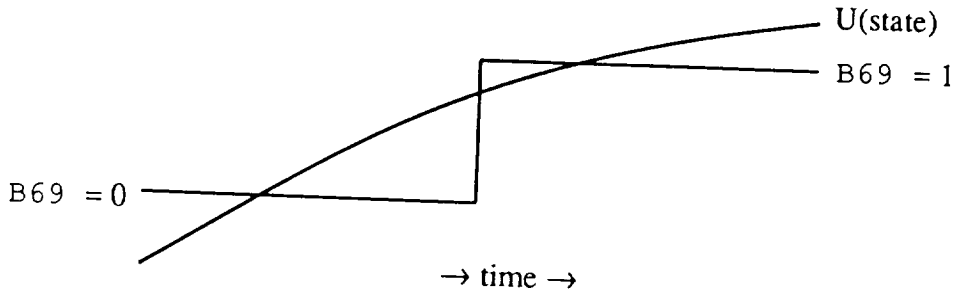

Figure 3.10: Noise bits that change slowly relative to estimated state values look relevant.

C.5 Discrete Reinforcement: The D Algorithm

We found the standard Q technique insufficiently sensitive in the Amazon domain. The problem is that Q simply sums all the reinforcement it gets, without distinguishing between different reinforcement values. For example, if the system is acting at random, as it does initially, it will typically have to shoot off many shuriken before killing a ghost. As the value of killing a ghost is only 10 and the cost of shooting is -.1, the 10 can get lost when summed with enough -.1s. To solve this problem, we extended Q to make more distinctions. Specifically, we effectively added

a third dimension to $Q(i,a)$, keeping track of $D(i,a,r)$, the discounted future probability of receiving reinforcement r after performing action a given input i:

$$D(i(t),a(t),r) = \sum_{k=0}^{\infty} \gamma^{t+k} p(r(t+k) = r.$$

This extension separates out the various possible reinforcement values and so gives better statistical information. The Q values can be recovered with the identity

$$Q(i,a) = \sum_{r \in R} r\, D(i,a,r)$$

where R is the space of reinforcements. This extension to Q-learning is possible only when the reinforcement given is discrete and takes on only a relatively small number of values (though it might be possible to use buckets to apply it in cases of continuously varying reinforcement).

It's not surprising that D-learning works better than Q-learning; it is superior for the same reason that Q-learning is better than the adaptive heuristic critic: it keeps track of more distinctions. The logical next step in this progression would be to keep track of input-action-input triples, as (for instance) Drescher [69] has done; we'll have more to say about this in section 3.2.E.

D Performance

It's hard to know how to evaluate the learning rate of G, lacking obvious alternatives to compare it with. The system learns many times slower than Q on simple problems, because it has to repeatedly split, throw away its old statistics, and start over again. However, we have successfully run G on problems with thirty input bits, for which Q could not allocate memory to store its table, much less accumulate the billions of ticks of experience necessary to fill it out. The only current alternative means of generalization for TD learning is backpropagation. A fair comparison of the two would require a spectrum of domains, as the performance of TD plus backpropagation is unpredictable.

The relevance-splitting component of G has performed very well in our problem domain. We have run it for well over a million ticks with ten bits of noise given as inputs in addition to the standard inputs specified earlier; it never split on any of these noise bits. On the other hand, on several runs each of several variations of the problem and it has always split on all the bits that *are* relevant.

The Amazon problem we tested G on had only six (relevant) bits of input, whereas the domain used by Lin's tests of Q plus backpropagation has 145, most or all of which are probably relevant to action in every situation. G would not work on such a problem; it would try to split on every bit and soon generate too large a tree. However, Lin's encoding of the problem is essentially retinotopic. As Chapman has argued [51], the inputs to mammalian learning systems are almost certainly not retinotopic, and we should not try to optimize our learning systems for such inputs. He hypothesizes that "intermediate" visual inputs will be much easier to learn from than retinotopic ones.

E Conclusions and Future Directions

Our first conclusion from this work is that temporal difference learning is not yet a mature technology that can be readily applied to specific learning tasks (such as that of learning visual routines). The bulk of experience with TD methods has been in simple domains that do not stretch their capabilities. Further experimentation in complex domains will probably uncover further difficulties and inspire further extensions to the basic algorithm.

This paper was concerned with the input generalization problem. A similar problem arises with output generalization: if the number of actions is very large, the learner can not hope to try each in every state. For example, our target visual system has several dozen control bits that the action policy must set on every tick. Kaelbling has described an approach to this problem in previous work [57]. We believe, however, that the G algorithm should be directly applicable to the output generalization problem. That is, the system could keep track of the effect of individual output bits on reinforcement received in particular input blocks, and construct a tree of output bit relevance analogous to the input bit relevance tree. We have not implemented this, however.

Finally, we conclude that temporal difference methods (however amended) will simply be inadequate to learn effective action policies in some domains. TD methods are inherently single-tick based: they consider only the value of actions taken in a state. Considering pairs of states allows learning of state-transition models for actions. Such models enable planning methods for getting to desirable states. Such methods have been advocated to improve the performance of an agent once the models have been learned [58, 59, 70, 71]. We observe, however, that they may be even more important in learning the models in the first place. The G algorithm, in learning the Amazon problem, spends almost all of its time in uninteresting, already-overlearned portions of the state space. It would learn immensely faster if it were able to deliberately enter parts of the state space that it didn't know enough about.[7] *Efficient learning requires effective experimentation, which requires (at least) action transition models.* This will be increasingly true in domains in which the most valuable states are ones that the learner is unlikely to come upon at random.

Such multiple-tick policy learning, however, poses severe combinatorial problems. The number of possible state transitions is enormous; another dimension of generalization is required. Furthermore, we are unsure how a system could effectively exploit the information if it were learned; the usual combinatorial arguments against classical planning apply. Three previous studies from the literature are relevant. Sutton [59] describes a system which successfully combines transition learning and planning with Q-learning; however, he tested the combination only in a tiny domain in which the combinatorics could be expected to be tractable. Lin [58] made a similar combination in a more complex domain, using a backpropagation network to represent the state transition information. However, the connectivity of this network was chosen to exploit the locality of the domain, and the resulting system is not general purpose. Finally, Drescher's marginal attribution method [69] learns action transition models by a method in some respects similar to the G algorithm. (In fact G was partly inspired by the marginal attribution method.) However, he reports only experiments in a small domain and does not analyze the combinatorics, which look bad, in depth. Nonetheless, we are forced to conclude that some method of this sort, with appropriate means of reducing the combinatorics, is required.

7. The IE algorithm doesn't help with this: IE chooses poorly-understood actions in whatever state it finds itself in, but does not try to get itself into states in which there are poorly-understood actions.

3.3 Gaining Information Through Language

There has been much controversy recently as to whether the rules of interaction discovered by conversation analysts are amenable to use by computers [72, 73, 74]. Button [75] has argued that the rules of conversation are of a different ontological category than the rules used by computers, and that this means computers cannot be programmed to engage in conversation. Others [76, 77, 78] have argued to the contrary that the rules of conversation can be captured in a program, and indeed that some have been. I will argue for a third position. Button is right in his critique of existing attempts to import conversation analysis into computational linguistics and in his argument that there is a rule type mismatch. His arguments do not, however, show that computers cannot in principle be programmed to engage in conversation. I will argue by analogy to computer network protocols that an *interactionist* computational interpretation of the conversation analytical rules is possible, and that Button's critique can thereby be bypassed.

Button [79] has argued that computers cannot engage in conversation because the rules of computation are of a different sort than the rules of conversation. The rules (or programs) that govern computers

- are explicitly represented,
- are causally efficacious, directly engendering the activities they describe, so that
- they cannot be violated, and thus
- have the force of mathematical laws.

The rules of conversation that have been discovered by conversation analysts [80], on the other hand,

- are typically not represented by their users,[8]
- are not causally efficacious, but nevertheless
- apply uniformly, even when they are violated, and have the force of social norms.

If these properties seem odd, let us consider some examples. On the first point, most of us are unaware of the role that gaze direction plays in the selection of next speaker in three-way conversations [81]. A conversation analyst would argue that though it is logically possible that we unconsciously represent rules about gaze direction, there is no evidence for that. The other three points may be illustrated by the rule that you should shake the hand of a person you are introduced to. This rule is not a physical law; you are able to violate it at will. Nevertheless, the rule applies even when you have violated it: the person you have been introduced to is liable on the basis of the rule to consider you rude. You are, then, liable to be held to account for the violation; you may get an unfortunate reputation. Thus the normative character. These four properties are summarized by conversation analysts in two terms: people are said to *orient to* rules, rather than being governed by them, and rules are viewed as a *resource* in conversation, not a determining factor.[9]

Button argues that the incompatibility between these sorts of rules implies (1) that recent attempts to import rules from conversation analysis into computational linguistics are misguided, and (2) that computers cannot, in principle, participate in conversation. I believe he is right on the first count and wrong on the second.

The book *Computers and Conversation* [74] describes several systems [77, 78] that attempt to incorporate rules taken from conversation analysis into natural language interface systems. In

8. Once discovered, they may be represented by conversation analysts, who also of course use them.

9. For one attempt to explicate these ideas in an AI context, see [82].

these systems, conversational rules, such as those of turn taking, are explicitly represented as datastructures in a grammar or formal rule system and are used as the basis of a natural language processing program. Since these rules are explicit, causally govern action, and cannot be violated, they are of a quite different character than the conversation analytical rules which inspired them.

Does this matter? It depends on one's purposes. For building better human/computer interfaces, this transformation in rule type will be perfectly justified if indeed it results in interfaces that are easier to use than ones not inspired by conversation analysis. This is an engineering question, not a foundational one, and it can only be answered empirically, not analytically. If, however, one wishes to better understand human interaction by computational modeling, the transformation is indeed troubling. The four properties of conversational rules seem to be central characteristics of human action more generally [83]. We are, as Button says, "going up a blind alley" if we continue to ignore this mismatch in rule type.

The easiest response to this difficulty is to suggest, as Hirst [73] has, that "Button seems to be saying nothing more than that [conversation analytical] rules must be represented declaratively, not procedurally, so that they can be deliberately and explicitly invoked with a view to their effect. But far from precluding their use by a computer, this argument suggests that they fit very nicely into present-day AI reasoning systems!" This approach has been pursued by Fraser and Wooffitt [76]. They propose that conversational rules are explicitly represented and manipulated by metarules [84]. The metarules can choose to violate base-level rules under appropriate circumstances. Thus the base-level rules are not causally efficacious and do not directly determine action. They can, however, be used when violated to explain another agent's actions.

However, this valiant attempt fails to capture the conversation analytical notion of rule. First, Fraser and Wooffitt's rules are still representations. The conversation analytical perspective is not that rules should be represented declaratively, but that they should not be represented *at all*.[10] Second, though individual rules in the implemented system are not causally efficacious, the set of them is; the logic of the group of them cannot be violated. Third, there is no account of the sense in which the rules have the force of social norms.

We have thus far considered Button's first claim, that the type mismatch between conversational and computer rules means that direct translation of the former into the latter falsifies their nature. Let us now consider his second claim, that this implies that computers cannot engage in conversation.

Button hasn't shown that computers *can't* orient to rules, just that in current AI practice they *don't*. To see why Button's objections need not be fatal, we need to understand the interactionist perspective of conversation analysis, and to see how this perspective might be interpreted computationally.

Although the subject matter of conversation analysis is roughly the same as that of computational linguistics, the goals of the two fields are fundamentally different. Conversation analysis does not seek explanations of linguistic behavior. It is concerned rather with describing patterns of interaction between people. Because it is not looking for causal explanations, and because it is concerned with inter-individual phenomena, it is not concerned with things-in-the-head such as representations. Conversation analysis does not deny that there are things-in-the-head; it is simply uninterested because they are seen as irrelevant to its goals. It is, thus, not part of cognitive science, and what counts as progress in each discipline does not look like progress to the other.

10. Hirst's confusion on this point is understandable; Button's exposition does not make the point explicit.

These ideas may be easiest to understand by way of an analogy. Consider a computer workstation running a network file system that lets you access files on a remote file server. The client and server communicate via a network file protocol such as FTP or NFS. This protocol is a set of rules that specify how the client and server are to interact. However, the protocol does not appear in the network file system implementation: it isn't a datastructure, procedure, set of procedures, or any other thing-in-the-computer. It is merely a specification. In fact, the computer does not represent the protocol it uses. That's probably just as well, because representing and manipulating the protocol – as a set of first-order axioms in a theorem prover, for instance – would be a difficult and computationally inefficient way to build a network file system.

Having concluded that the protocol is not in the computer, should we suppose instead that it is in the environment? Or, by analogy, having abandoned the mentalist supposition that patterns of action result from representations of those patterns, must we accept the behaviorist supposition that action is patterned by stimuli in the environment?

The environment of a computer on the net is another computer on the net. But if the protocol is not in the one, it is not in the other either, so that doesn't help any. The protocol *is* written down in a natural language document called an RFP; but that doesn't play any role in the actual operation of the file system. The protocol may also be represented in the head of the file system's writer. These representations do play a causal (because historical) role in the operation of the network code; but not in the usual sense in which representations play a role in action in AI. The representations in the designer's head can change (he may forget the protocol) without affecting the operation of the network code.

To return to the conversational rules case, the observation that the network protocol is in the head of the programmer is irrelevant, because there's no programmer in whose head the conversation analytical rules would live. Similarly, although network protocols are typically written down on paper, the rules of conversation mostly aren't because they mostly haven't been discovered yet.

Thus, the interactionist perspective of conversation analysis sidesteps the mentalism-behaviorism debate.[11] For conversation analysis, the phenomena of interest can be located neither in the head nor in the environment. Like network protocols, they are interactional. To understand *how* people do what they do, one has to know about things-in-the-head; but conversation analysis refuses to speculate about these, because it is interested only in *what* people do. Thus, for conversation analysis, rules are not causal agents, but descriptions of regularities in interaction. By analogy, one might observe network traffic and induce the structure of a protocol without any access to the programs that use the protocol. Indeed, in some cases this access might not help much; it is notorious that the *uucp* protocol is undocumented and very hard to induce by reading the convoluted code that uses it.

We can now diagnose the problems with existing computational interpretations of conversation analysis as symptomatic of a deeper problem: the systems retain a mentalist orientation, with their designers seeking to locate explanations of action in mental structures. This orientation is endemic in AI generally; but nothing precludes interactionist AI in principle [82]. In fact, the network protocol analogy suggests starting points for a different computational approach to interpreting conversation analysis. We'll see now, in another example, how a rule governing computer network communication has three of the four properties of conversation analytical rules cited earlier: it is not represented, it is not causally efficacious, and it applies even when it is violated.

---

11. For a clear exposition of how a third alternative to mentalism and behaviorism is possible, see [85].

The fundamental rule of communication on an Ethernet local network is that *only one computer may transmit at a time*. If two computers try to talk at once, there is a "collision" and the messages are scrambled. However, such collisions are unavoidable and occur regularly. When a collision occurs, the transmitting computers detect it and engage in a "retransmit protocol" to rectify the problem.

This rule has all the cited properties of conversation analytical rules except being a social norm. First, as with the file protocol, the Ethernet controller does not anywhere represent or otherwise include the rule. The rule is too pervasive and fundamental a feature of the situation to require representation.

Second, representing the rule wouldn't be useful, in any case, because it is not causally efficacious; it *cannot* engender the constraint it imposes. The problem is that the rule is a constraint on global interaction, not on individual action. A computer does not know when another computer is about to transmit, so it can't avoid collisions. The rule "there shall be no collisions" could be enforced by a complex protocol which gave machines information about when other machines might start transmitting. But such protocols require significant design and don't just fall out of the interactional rule. Thus we see that representing an interactional rule is not always a help in conforming to it.

As for the third property, the Ethernet collision rule applies even when it is violated. When two computers do transmit simultaneously, the rule is used to interpret the resulting garble on the network, and the retransmit protocol is used to repair the trouble.

Thus the protocol is an interactionist (rather than mentalist) form of rule use, but it is undeniably computational. What, then, about conversation analysis and AI? Button is right that the conversation analytical rules should not be represented as expert-system-like rules. But the fact that computers are *governed* by one sort of rules (programs) does not preclude their *orienting to* another sort (such as those of conversation analysis). Does the fact that the rules of conversation are not represented mean that we must eschew Lisp and use holistic neural networks? No. There's nothing mystical about the guts of a network file system: procedures manipulate datastructures representing packets and connections and host addresses. Yet the program uses a protocol it does not represent.

Of course network communication is in almost all other respects unlike human conversation; it would be wrong to suggest that Ethernet controllers orient to the no-collisions rule. But this example suggests that if the fourth issue – the normative character of rules – were addressed, Button's argument may not hold water. I think this, and not the representational issue, is the hard and interesting challenge of conversation analysis for computational linguistics.

What does it mean that the rules of conversation have the force of social norms? I doubt that there can be a general answer to this question. Conversation analysts, following Garfinkel's ethnomethodological critique of the appropriation of common-sense categories like "social norm" as theoretical terms [86, 80], would not even attempt to answer it. However, some elementary observations may point in the right direction. First, social action is *accountable* in the sense that one may be required to produce an account of the rationality of one's action. This requirement is relatively unproblematic; it could be argued that some existing AI systems produce such accounts. Second, when social interaction runs into trouble, as it regularly does, the participants are required to make an effort to find the location of difficulty, to determine which participant is responsible for the problem, and to take steps to repair it. Third, this process of trouble location and repair is not a mechanical one; it requires interactive work and a commitment to negotiating the specifics of concrete marginal cases.

I believe it is possible to build a natural language system whose rule use satisfies the first three

criteria in the same way the Ethernet controller does; and whose action is arguably subject to social norms in virtue of producing accounts, repairing misunderstandings, and negotiating assignment of the location of difficulties. This would not show that computers can engage in conversation; there are many other obstacles. It would, however, demonstrate that the particular problems Button raises are not the stumbling blocks.

4 Conclusions

This report has described work done on the subject of acting to gain information. This work included theoretical studies on the foundations of perception and action in intelligent, embedded agents, as well as work on information-gathering in three qualitatively different modes: visual perception, reinforcement learning, and conversational interaction.

In our theoretical investigations, we examined mathematically the ways in which information and control interact, including the feasibility of different synthesis methodologies for handling goals of information acquisition. Through these investigations, the interdependence of perception and action were elucidated in the situated-automata framework. It was shown how in certain circumstances action strategies could, in principle, be synthesized automatically from descriptions of the environment and task requirements. It was also shown under what circumstances the perception and action components are, in fact, interdependent and must be co-specified. Suggestions were offered on practical development methodologies.

Our work on visual perception was motivated to a large extent by the felt need in the intelligent-agent community for high-level interpretations of visual sensor data. Our approach has been to bring the process of visual sensing and interpretation under the control of a reactive, goal-directed control system. We defined primitive measurements that were amenable to such control, along with an architecture that was capable of supporting constant-time updates of rich, hierarchical object representations. Experiments were carried out to demonstrate several advances in recognition and tracking.

In the area of learning, we carried out investigations aimed at improving the performance of reinforcement learning algorithms by maintaining statistics that identified relevant inputs so that input generalization could be effectively performed. Experiments were carried out that demonstrated promise. Finally, in the area of information gain through conversational interaction, we explored the nature of computational rules and the role of convention in facilitating information pickup in dialogue.

Our research has uncovered numerous opportunities for continued exploration. Theoretical issues to be studied include advanced decomposition techniques for representations involving statistical information, including extension of the formal analysis of Percm-like schemata for perceptual updates based on statistical recognition. Our work on visual perception suggests a long list of fruitful projects aimed at widening the set of measurement types employed by the system, extending our techniques for finding homogeneous regions beyond the single-modality (color) and connected-component approach used in this study, and extending our recognition techniques and testing them against a larger set of models. Additional algorithmic ideas are also required to recognize temporally extended visual events. In the area of learning, one useful direction for further research would be the development of a principled approach to the construction of hybrid systems, where directly programmed action rules would co-exist with autonomously learned functionality in a single, unifying framework. Finally, in the area of conversational interaction, simulations should be developed to illustrate and study coordination phenomena enabled by linguistic exchange.

The development of a comprehensive body of theory and practice on acting to gain information will have significant impact on a wide range of applications but will undoubtedly involve many years of research and interactions across a variety of disciplines.

Bibliography

[1] Stanley J. Rosenschein and Leslie Pack Kaelbling, "The Synthesis of Digital Machines with Provable Epistemic Properites," in Joseph Y. Halpern, editor, Theoretical Aspects of Reasoning About Knowledge:

[2] Stuart Russell and Eric Wefald, "Do the Right Thing: Studies in Limited Rationality," The MIT Press, 1991.

[3] Thomas L. Dean and Michael P. Wellman, "Planning and Control," Morgan Kaufmann, 1991.

[4] Peter Ramadge and Murray Wonham, "The Control of Discrete Event Systems," Proceedings on the IEEE, 77(1):81-89, 1989.

[5] Leslie Pack Kaelbling and Stanley J. Rosenschein, "Action and Planning in Embedded Agents," Robotics and Automation, 6 (1990).

[6] N. Wiener, "Cybernetics: or Control and Communication in the Animal and the Machine," MIT Press, 1969.

[7] Stanley J. Rosenschein, "Synthesizing Information-Tracking Automata from Environment Descriptions," Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, pages 386-393, Morgan Kaufmann, 1989

[8] M. Swain and M. Stricker, "Promising Directions in Active Vision", NSF Active Vision Workshop (written by attendees of workshop), U. of Chicago, 1991.

[9] R. Bajcsy, "Active perception," Proceedings of the IEEE, volume 76, pages 996-1005, 1988.

[10] P. Burt, "Smart sensing with a pyramid vision machine," Proceedings of the IEEE, number 76, pages 1006-1015, 1988.

[11] E. Krotkov, "Active Computer Vision by Cooperative Focus and Stereo," Springer Verlag, New York, 1989.

[12] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," International Journal of Computer Vision, number 1, pages 333-356, 1988.

[13] D. Ballard, "Animate vision," Artificial Intelligence, 48:57-86, 1991.

[14] L.H. Matthies, "Stereo vision for planetary rovers," Technical Report JPL D-8131, Jet Propulsion Laboratory, California Institute of Technology, January 1991.

[15] C. Brown and R. Nelson, "Image understanding research at Rochester", IUW 1990, pp. 76.

[16] M.J. Swain, "Color Indexing," TR 360, University of Rochester, Computer Science Dept., Rochester, NY, Nov. 1990.

[17] D. Coombs and T.J. Olson, "Real-time vergence control for binocular robots," IJCV, 1991.

[18] D. Coombs and C. Brown, "Real-time smooth pursuit tracking for a moving binocular robot", CVPR 1992, p. 23.

[19] J. Woodfill and R. Zabih, "Using motion vision for a simple robotic task," Sensory Aspects of Robotic Intelligence, Working Notes, AAAI Fall Symposium Series, Asilomar, California, November 1991.

[20] C.L. Fennema, Jr., "Interweaving Reason, Action and Perception", COINS TR91-56, University of Massachusetts at Amherst, September 1991.

[21] C. Hewitt, "The Challenge of Open Systems," Byte, vol. 10, no. 4, pp. 223-242, April 1985.

[22] J.C. Taenzer, "Some Psychophysical Limitations on Reading Performance," PhD thesis, Stanford University, June 1971, Stanford Electronics Laboratories TR No. 4828-5.

[23] H.K. Nishihara, "Intensity, visible-surface, and volumetric representations," Artificial Intelligence, 17:265-284, 1981.

[24] D. Marr and H.K. Nishihara, "Visual information processing: Artificial intelligence and the sensorium of sight," Technology Review, 81(1):28-49, 1978. Also in Readings in

Computer Vision: Issues, Problems, Principles, and Paradigms, edited by M.A. Fischler and O. Firschein, Morgan Kaufmann, Los Altos, 1987.
[25] B. Julesz, "Foundations of Cyclopean Perception," University of Chicago Press, 1971.
[26] H.K. Nishihara, "Hidden information in transparent stereograms," Proc. of the Twenty-First Asilomar Conf. on Signals, Systems, and Computers, pages 695-700, Pacific Grove, CA, Nov 1987.
[27] H.K. Nishihara, "Tests of a sign correlation model for binocular stereo," Investigative Ophthalmology and Visual Science, 30(3):389, 1989.
[28] S.B. Stevenson, L.K. Cormack, and C.M. Schor, "Depth attraction and repulsion in random dot stereograms," Investigative Ophthalmology and Visual Science, 30(3):390, 1989.
[29] D. Marr and T. Poggio, "A computational theory of human stereo vision," Proceedings of the Royal Society of London, 204:301-328, 1979.
[30] H.K. Nishihara, "Practical real-time imaging stereo matcher," Optical Engineering, 23(5):536-545, October 1984. Also in Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, edited by M.A. Fischler and O. Firschein, Morgan Kaufmann, Los Altos, 1987.
[31] H.K. Nishihara and P.A. Crossley, "Measuring photolithographic overlay accuracy and critical dimensions by correlating binarized Laplacian of Gaussian convolutions," IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 17-30, January 1988.
[32] N. Hunt, P.A. Crossley, and H.K. Nishihara, "Binocular stereo techniques for the direct measurement of area or volum," SPIE Conf. on Reconnissance, Astronomy, Remote Sensing, and Photogrammetry, volume 1070, Los Angeles, CA, Jan 1989.
[33] L.A. Sorayama, "Localization of depth edges in stereo images," Master's thesis, Massachusetts Institute of Technology, August 1984.
[34] G. Healey, "Segmenting Images Using Normalized Color", IEEE SMC, 22(1):64-73.
[35] J, A. van Dam Foley, S. Feiner and J. Hughes, "Computer graphics: principles and practice," Addison-Wesley, 1990.
[36] R. Duda, and P. Hart, "Pattern Classification and Scene Analysis," Wiley-Interscience, 1973.
[37] D. Ballard, and C. Brown, "Computer Vision," Prentice Hall, Englewood Cliffs, NJ, 1982.
[38] J.B. Burns, "Matching 2D images to Multiple 3D objects usingview description networks", Computer Science TR 92-17, University of Massachusetts, Amherst MA, 1992.
[39] J.B. Burns, "View variation of point-set and line-segment features", in IEEE PAMI, 15(1):51-68, 1993.
[40] Y. Lamdan, and H. Wolfson, "Geometric Hashing: a general and efficient model-based recognition scheme", Proc. ICCV, pp. 238-249, 1988.
[41] D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes", in Pattern Recognition, 13(2):111-122, 1981.
[42] T. Breuel, "Fast recognition using adaptive subdivisions of transformation space", in Proc. CVPR, pp. 445-451, 1991.
[43] R. J. Baron, "Mechanisms of human facial recognition," Int. J. Man-Machine Studies, 15, pp. 137—178, 1981.
[44] S. R. Cannon, G. W. Jones, R. Campbell, and N. W. Morgan, "A computer vision system for identification of individuals," Proc. IECON, Vol. 1, pp. 347-351, 1986.
[45] G. W. Cottrell and J. Metcalfe, "EMPATH: Face, gender and emotion recognition using holons," In R.P. Lippman, J. Moody, and D.S. Touretzky (Eds.), Advances in neural information processing systems 3, San Mateo, CA: Morgan Kaufmann.
[46] M. Turk and A. Pentland, "Face recognition using eigenfaces," Proc. CVPR, Maui, Hawaii, pp. 586-591, June 1991.
[47] W. W. Bledsoe, "Man-machine facial recognition," Panoramic Research Inc., Palo Alto, CA, Rep. PR1:22, Aug. 1966.

[48] T. Kanade, "Picture processing system by computer complex and recognition of human faces," Dept. of Information Science, Kyoto University, Nov. 1973.
[49] M. Turk, "Interactive-Time Vision: Face Recognition as a Visual Behavior," Ph.D. Thesis, The Media Laboratory, Massachusetts Institute of Technology, Sept. 1991.
[50] Richard S. Sutton, "Learning to Predict by the Method of Temporal Differences," Machine Learning, 3:1 (1988), pp.9-44.
[51] David Chapman, "Vision, Instruction, and Action," MIT Press, 1991. Revised version of MIT AI Laboratory Technical Report 1204, April 1990.
[52] David Chapman, "Intermediate Vision: Architecture, Implementation, and Use," Teleos Research TR-90-06, October, 1990.
[53] Steven D. Whitehead and Dana H. Ballard, "Active Perception and Reinforcement Learning," University of Rochester Computer Science Department Technical Report 331, 1990. To appear in Machine Learning.
[54] A.G. Barto, R.S. Sutton, and C.J.C.H. Watkins, "Learning and Sequential Decision Making," University of Massachusetts Department of Computer and Information Science Technical Report 89-95, Amherst, Massachusetts, 1989.
[55] Richard S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," PhD Thesis, University of Massachusetts at Amherst, 1984.
[56] Christopher John Cornish Hellaby Watkins, "Learning from Delayed Rewards," PhD Thesis, King's College, Cambridge, 1989.
[57] Leslie Pack Kaelbling, "Learning in EmbeddedSystems," Teleos Research TR-90-04, June 1990.
[58] Long-Ji Lin, "Self-improving Reactive Agents: Case Studies of Reinforcement Learning Frameworks," Carnegie Mellon University Technical Report CMU-CS-90-109, 1990. Also to appear in Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats.
[59] Richard S. Sutton, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, Morgan Kaufmann, 1990.
[60] Charles W. Anderson, "Strategy Learning with Multilayer Connectionist Representations," Proceedings of the Fourth International Workshop on Machine Learning, Ann Arbor, Michigan, 1987, pp. 103-114.
[61] J. F. Shepansky and S. A. Macy, "Teaching Artificial Neural Systems to Drive: Manual Training Techniques for Autonomous Systems," Proceedings of the First Annual International Conference on Neural Networks, San Diego, CA, June 21-24, 1987.
[62] Randall D. Beer, "Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology," Academic Press, San Diego, 1990.
[63] Rodney A. Brooks, "A Robot that Walks: Emergent Behavior from a Carefully Evolved Network," Neural Computation, 1:2, Summer, 1989.
[64] Jonathan Hudson Connell, "Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature," Academic Press, San Diego, 1990.
[65] Leslie Pack Kaelbling, "Goals as Parallel Program Specifications," Proceedings of the Seventh National Conference on Artificial Intelligence, Minneapolis-St. Paul, Minnesota, 1988.
[66] Leslie Pack Kaelbling and Stanley J. Rosenschein, "Action and Planning in Embedded Agents," Robotics and Automation, 6 (1990).
[67] George W. Snedecor and William G. Cochran, "Statistical Methods," Iowa State University Press, Ames, Iowa, eighth edition, 1989.
[68] George W. Snedecor and William G. Cochran, "Statistical Methods," Iowa State University Press, Ames, Iowa, eighth edition, 1989.
[69] Gary Drescher, "Made-up Minds: A Constructivist Approach to Artificial Intelligence," MIT Press, 1991, revised version of PhD Thesis, Department of Electrical Engineering and Computer Science, MIT, 1989.
[70] Richard S. Sutton and Brian Pinette, "The Learning of World Models by Connectionist

Networks," Proceedings of the Seventh Annual Conference of the Cognitive Science Society, 1985, pp. 54-64.

[71] Steven D. Whitehead and Dana H. Ballard, "A Role for Anticipation in Reactive Systems that Learn," Proceedings of the Sixth International Workshop on Machine Learning, Ithaca, New York, Morgan Kaufmann, 1989, pp. 354-357.

[72] Nigel Gilbert, ed., "Proceedings of the AAAI Workshop on Complex Systems, Ethnomethodology and Interaction," Boston, July 1990.

[73] Graeme Hirst, "Does Conversation Analysis Have a Role in Computational Linguistics?," Computational Linguistics, 17:2 (June 1991) pp. 211-227.

[74] Paul Luff, Nigel Gilbert, and David Frohlich, eds., "Computers and Conversation," Academic Press, London, 1990.

[75] Graham Button, "Going Up A Blind Alley: Conflating Conversation Analysis and Computer Modeling," in Paul Luff, Nigel Gilbert, and David Frohlich, eds., Computers and Conversation, Academic Press, London, 1990, pp. 67-90.

[76] N.M. Fraser and R.C. Wooffitt, "Orienting to rules," in Nigel Gilbert, ed., Proceedings of the AAAI Workshop on Complex Systems, Ethnomethodology and Interaction, Boston, July 1990, pp. 69-80. Revised version to appear as "The dynamics of rule use," in Journal of Intelligent Systems.

[77] David Frohlich and Paul Luff, "Applying the Technology of Conversation to the Technology for Conversation," Chapter 9 in Paul Luff, Nigel Gilbert, and David Frohlich, eds., Computers and Conversation, Academic Press, London, 1990, pp. 187-220.

[78] Nigel Gilbert, Robin Wooffitt, and Norman Fraser, "Organizing Computer Talk," Chapter 11 in Paul Luff, Nigel Gilbert, and David Frohlich, eds Computers and Conversation, Academic Press, London, 1990, pp. 235-257.

[79] Graham Button, "Going Up A Blind Alley: Conflating Conversation Analysis and Computer Modeling," in Paul Luff, Nigel Gilbert, and David Frohlich, eds., Computers and Conversation, Academic Press, London, 1990, pp. 67-90.

[80] John Heritage, "Garfinkel and Ethnomethodology," Polity Press, Cambridge, England, 1984.

[81] Charles Goodwin, "Restarts, Pauses, and the Achievement of a State of Mutual Gaze at Turn-Beginning," Chapter 9 in Don Zimmerman and Candace West, eds., "Language and Social Interaction," Sociological Inquiry, 50:3-4}, 1980, pp. 272-302.

[82] Philip E. Agre and David Chapman, "What are Plans For?," in Robotics and Automation, 6 (1990) pp. 17-34. Also in Pattie Maes, ed., "Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back," MIT Press, Cambridge, Massachusetts, 1990.

[83] Hubert Dreyfus, "What Computers Can't Do," Harper and Row, New York, 1979.

[84] bibitem{Davis} Randall Davis, "Meta-rules: Reasoning about Control," Artificial Intelligence, 15 (1980), pp 179-222.

[85] Beth Preston, "Behaviorism and Mentalism: Is There A Third Alternative?," to appear, "Synthese."

[86] Harold Garfinkel, "Respecification: evidence for locally produced, naturally accountable phenomena of order*, logic, reason, meaning, method, etc. in and as of the essential haecceity of immortal ordinary society, (I)—an announcement of studies," Chapter 2 in Graham Button, ed., "Ethnomethodology and the Human Sciences," Cambridge University Press, Cambridge, 1991, pp. 10-19.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 JUL 93	Final Report 1/91-7/93

4. TITLE AND SUBTITLE
Acting to Gain Information

5. FUNDING NUMBERS
NAS2-13326

6. AUTHOR(S)
Stanley J. Rosenschein, J. Brian Burns, David Chapman, Leslie P. Kaelbling, Philip Kahn, H. Keith Nishihara, Matthew Turk

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Teleos Research
576 Middlefield Road
Palo Alto, CA 94301

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-93-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NASA Ames Research Center
Moffett Field, CA 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report is concerned with agents that act to gain information. In previous work, we developed agent models combining qualitative modeling with real-time control. That work, however, focused primarily on actions that affect physical states of the environment. The current study extends that work by explicitly considering problems of active information-gathering and by exploring specialized aspects of information-gathering in computational perception, learning, and language.

In our theoretical investigations, we analyzed agents into their perceptual and action components and identified these with elements of a state-machine model of control. The mathematical properties of each was developed in isolation and interactions were then studied. We considered the complexity dimension and the uncertainty dimension and related these to intelligent-agent design issues.

We also explored active information gathering in visual processing. Working within the active vision paradigm, we developed a concept of "minimal meaningful measurements" suitable for demand-driven vision. We then developed and tested an architecture for ongoing recognition and interpretation of visual information. In the area of information gathering through learning, we explored techniques for coping with combinatorial complexity. We also explored information gathering through explicit linguistic action by considering the nature of conversational rules, coordination, and situated communication behavior.

14. SUBJECT TERMS
Intelligent Agent, Situated Automata, Active perception, Reactive planning and control

15. NUMBER OF PAGES
67

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited